University of Alberta

New Techniques for *p*-Cycle Network Design

by

Anthony Conrad Sack ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**

Department of Electrical and Computer Engineering

Edmonton, Alberta

Fall 2004

Library and
Archives Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 0-612-95845-0
Our file Notre référence
ISBN: 0-612-95845-0

The author has granted a non-exclusive license allowing the Library and Archives Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou aturement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

"By Endurance We Conquer"

- Shackleton family motto

To those with the courage to change.

Acknowledgements

There are a lot of people who helped me over the last few years, not only with the research and writing for this thesis, but also in a more general sense by encouraging my ongoing growth and development. I am really grateful to all of them.

In a personal sense, I would like to first thank **Pauline Atwood** for being a supportive friend and for her encouragement and inspiration. I also very much appreciate the advice and support of my good friend and former classmate **Erica Povhe**.

For helping me learn about communication (in a different sense than survivable networking) in a warm and supportive environment, I thank everyone in the Fun Speakers Toastmasters Club, and in particular, **Linda White** and **Tim Lambert** for teaching me about leadership, and **Mich Wilke** for being such a great friend and colleague.

For financial assistance, I thank **Wayne Grover** and our funding agency **NSERC**, **Karen and Gerald Sack**, and most especially **TRLabs** (which provided facilities and computing resources as well).

I thank the members of my committee, **Bruce Cockburn** and **Mike MacGregor**, for their comments and suggestions that helped improve the quality of the manuscript. As a collaborator, **François Blouin** at Nortel Networks helped make the mixed design study in Chapter 5 a really enjoyable project, and I appreciated all his ideas and advice.

The pleasant and friendly environment at TRLabs was in large part due to the warmth of people like **Dave Clegg**, **Rhoda Hayes**, **Faith Goller**, **Roger Pederson**, and **Tara Heron**. Another one of these people was **Linda Richens**, who always amazed me with her cheerfulness and willingness to help on so many occasions. A big thank-you to her.

It is hard to imagine a group of more talented and enthusiastic people than my colleagues in the Network Systems group. It was really a pleasure working with everyone and I know I learned something new from all of you. Special thanks go to **Matthieu Clouqueur**, **Dion Leung**, **Gangxiang Shen**, **Govind Kaigala**, and **Chioma Ezema**, who were here for much of the time that I was and were generous with both technical advice and general encouragement. I was fortunate to be in the office across from **Adil Kodian** and enjoyed our many conversations and the path-length project we worked on together that appears in Chapter 6. I appreciate all the technical and computing help he gave me and also simply the chance to work together because he is really a genuinely nice guy.

In some senses acting almost as a junior supervisor, and certainly as a great colleague and friend, **John Doucette** was very generous with his time and expertise and had a big impact on my development. He was always available whenever I had questions and would often drop by my office to say hello or arrange various social outings for the people in the lab. A big thank-you to **John** as well.

From working with **Wayne Grover**, I came to appreciate not only his technical expertise, but also a number of other qualities like diplomacy, enthusiasm, and an incredible work ethic. **Wayne** devoted a lot of time to my education and development, and as my supervisor gave me a lot of great ideas and feedback on our research questions, experiments, and the writing and presentations that followed. I am also grateful for the many philosophical discussions we had and the wisdom he conveyed. **Wayne** was a great mentor with both patience and enthusiasm, and I thank him for all his help and support.

Anthony Sack
September 16, 2004

Table of Contents

Chapter 1. Introduction **1**
- 1.1. Importance of Transport Networking 1
- 1.2. Synopsis and Highlights 3

Chapter 2. Background Concepts **6**
- 2.1. Optical Transport Networks 6
  - 2.1.1. Technologies and Basic Definitions 6
  - 2.1.2. Network Demands 7
  - 2.1.3. Failures and Survivability 8
  - 2.1.4. Nodal Degree, Redundancy, and Typical Networks 9
- 2.2. Protection and Restoration Techniques 11
  - 2.2.1. Automatic Protection Switching 12
  - 2.2.2. Ring Protection 14
  - 2.2.3. Mesh Restoration 15
  - 2.2.4. Ring-Mesh Hybrids 16
  - 2.2.5. Shared Backup Path Protection 17
- 2.3. Optimization and Integer Linear Programming 19
  - 2.3.1. General Background 19
  - 2.3.2. Problem Formulation 20
  - 2.3.3. Linear vs. Integer Programming 21
  - 2.3.4. Practical and Scientific Benefits of ILPs 22
  - 2.3.5. ILPs for Network Design 23

Chapter 3. What Are *p*-Cycles? **26**
- 3.1. Introduction to *p*-Cycles 26
- 3.2. Spare Capacity Preconnection 29
- 3.3. Cycle-based Preconfiguration 30
  - 3.3.1. Discovery of Preconfigured Cycles 30
  - 3.3.2. Method for Distributed Preconfiguration 31
  - 3.3.3. Associated Patents 31
- 3.4. Subsequent Initial Developments 32
  - 3.4.1. First *p*-Cycle Publications 32
  - 3.4.2. Node-Protecting and IP Layer *p*-Cycles 32
- 3.5. Refinements and Dissemination 33
- 3.6. WDM Considerations 33

3.6.1. Wavelength Conversion Effects and Initial Studies 34
3.6.2. Partial Wavelength Conversion 35

3.7. Hamiltonian Considerations........36
3.7.1. Hamiltonian Cycle Protection (HCP) and Related Methods 36
3.7.2. Restricted *p*-Cycle (RPC) Method 37
3.7.3. Hamiltonians in Semi-Homogeneous Networks........38

3.8. Cycle Preselection and Cycle-Finding Heuristics........38
3.8.1. Preselecting Candidate Cycles for an ILP Solution........38
3.8.2. Purely Algorithmic Methods for *p*-Cycle Network Design........40

3.9. Ring Mining........40

3.10. Extensions for Path Segment Protection........41

3.11. Further *p*-Cycle Ideas........42
3.11.1. Directed and Non-Simple *p*-Cycles 42
3.11.2. Forcers and the Protected Working Capacity Envelope 43
3.11.3. Dual-Failure Survivability 44
3.11.4. Mixed Design Approach with Other Methods........45
3.11.5. MPLS Layer *p*-Cycles........45

3.12. Overview Papers and Comparative Studies 45
3.12.1. General Comparison Articles........45
3.12.2. Confusion of *p*-Cycles with Similar Methods 47
3.12.3. Additional Overview Presentations 47

3.13. Summary........48

Chapter 4. Hamiltonian *p*-Cycles: Addressing Some Confusions, Providing New Insights........49

4.1. General Background 49

4.2. Role of Hamiltonian *p*-Cycles in a Capacitated Design 50
4.2.1. Conventional *p*-Cycle Design Model........50
4.2.2. Hop-Based Solutions 52
4.2.3. Distance-Based Solutions 55
4.2.4. Network Family Solutions and Hamiltonian Comparison........56

4.3. Hamiltonian *p*-Cycles in a Flat-Capacity Design 58
4.3.1. Hamiltonian Illustration and Calculation of Redundancy Lower Bounds 58
4.3.2. The *AE* Metric for Hamiltonians........61

4.4. Flat-Capacity Design in a Non-Hamiltonian Network 63

4.5. Semi-Homogeneous Networks Based on the *p*-Cycle Concept........65

4.6. Summary ... 67

Chapter 5. A Mixed Design Approach With *p*-Cycles ... 68

5.1. Motivation ... 68
  5.1.1. Differentiation of Service and Survivability Methods ... 68
  5.1.2. Introducing *p*-Cycles to the Mix ... 70
  5.1.3. Introducing Self Selection ... 70

5.2. Mixed Multi-Method Design Model ... 71
  5.2.1. Preliminary Details ... 72
  5.2.2. Multi-Method ILP Formulation ... 74
  5.2.3. Test Networks ... 78

5.3. Results for Assigned Protection Designs ... 79

5.4. Results for Self-Selected Designs ... 83

5.5. Generic "Real-World" Test Case ... 89
  5.5.1. Effect of Bi-criteria Objective ... 89
  5.5.2. Introduction of Modularity ... 90

5.6. Selection of a Single Best Pure Architecture ... 91

5.7. Summary ... 93

Chapter 6. Path Length Constraints for *p*-Cycles ... 94

6.1. Overview of Path Length Restrictions and Surrounding Issues ... 95
  6.1.1. Circumference vs. Hop Limits for *p*-Cycles ... 95
  6.1.2. Motivation for Examining Path Length Constraints ... 97
  6.1.3. The Potential Efficacy of Path Length Restrictions ... 98

6.2. ILP Optimization Model with Path Length Constraints ... 99

6.3. Experimental Setup and Results ... 104
  6.3.1. Mesh-Restorable Comparative Reference Designs ... 104
  6.3.2. Test Networks and Solution Method Details ... 105
  6.3.3. Results: Threshold Hop-Limit Effect ... 107
  6.3.4. Results: Comparing to Circumference-Limited Designs ... 109
  6.3.5. Results: Checking on a Non-joint, Non-modular Design ... 110
  6.3.6. Bi-criteria Technique ... 112

6.4. Summary ... 113

Chapter 7. Accurately Distributed Cycle Preselection ... 114

7.1. Motivation ... 114

7.2. The Hypothesis for Statistical Mimicry ... 115

7.2.1. Idea of Statistical Mimicry 115
7.2.2. Proposed Preselection Method 116
7.3. Model and Preselection Program 116
7.4. Test Methods and Results 118
7.4.1. Verification of Statistical Repeatability 118
7.4.2. Effect of Sample Size 120
7.4.3. Performance Over a Variety of Graphs 121
7.4.4. Performance Over Varying Average Nodal Degree 123
7.4.5. Discussion 124
7.5. Summary 126

Chapter 8. Conclusions 127
8.1. Main Contributions 127
8.2. Publications 129
8.3. Further Research Ideas 130
8.4. General Summary 131

Bibliography 132

Appendix A. Test Networks 146
A.1. Canada Network 146
A.1.1. Topology (from [101]) 146
A.1.2. Demand Matrix (Unit Demand Between Node Pairs) 146
A.1.3. Demand Matrix (Unit Demand Between *Adjacent* Node Pairs) 147
A.2. 20n40s1 Network 147
A.2.1. Topology (identical to "Net-A" in [87] (p. 11)) 147
A.2.2. Demand Matrix (Unit Demand Between Node Pairs) 148
A.3. Canada-A Network (variant of Canada) 150
A.3.1. Topology (all $c_j = 1$) 150
A.3.2. Demand Matrix (Unit Demand Between *Adjacent* Node Pairs) 150
A.4. COST 239-A Network (variant of COST 239) 150
A.4.1. Topology (all $c_j = 1$) 150
A.4.2. Demand Matrix (Unit Demand Between *Adjacent* Node Pairs) 151
A.5. 15n30s1-A Network (variant of 15n30s1) 151
A.5.1. Topology (all $c_j = 1$) 151
A.5.2. Demand Matrix (Unit Demand Between *Adjacent* Node Pairs) 152

- A.6. 15n30s1-B Network (variant of 15n30s1) 152
  - A.6.1. Topology (all $c_j = 1$) 152
  - A.6.2. Demand Matrix (Unit Demand Between *Adjacent* Node Pairs) 152
- A.7. COST 239 Network 153
  - A.7.1. Topology (from [103]) 153
  - A.7.2. Demand Matrix (from [58]) 153
- A.8. Generic U.S. Network (19n31s1) 154
  - A.8.1. Topology (from colleagues at Nortel Networks) 154
  - A.8.2. Demand Matrix (from colleagues at Nortel Networks) 155
- A.9. 13n23s Network (variant of 15n30s1) 155
  - A.9.1. Topology 155
  - A.9.2. Demand Matrix 155
- A.10. 15n26s1 Network (variant of 15n30s1) 156
  - A.10.1. Topology 156
  - A.10.2. Demand Matrix 156
- A.11. 12n19s Network (variant of 15n30s1) 157
  - A.11.1. Topology 157
  - A.11.2. Demand Matrix 157
- A.12. NSFNET Network 158
  - A.12.1. Topology (from [83]) 158
  - A.12.2. Demand Matrix 158
- A.13. 19n35s1 Network (related to 20n40s1) 159
  - A.13.1. Topology 159
  - A.13.2. Demand Matrix 160
- A.14. 40n80s1-60s Network 161
  - A.14.1. Topology 161
  - A.14.2. Demand Matrix 162

Appendix B. AMPL Models **166**

- B.1. *p*-Cycle SCP (implementation of model presented in [29],[40]) 166
- B.2. Multi-Method 1+1 APS, *p*-Cycle, and SBPP Design (Self-selected Case) (implementation of model in [21]) 167
- B.3. *p*-Cycle Modular JCP Design with Path Length Constraints (implementation of model in [110]) 171

List of Tables

Table 5.1. Typical restoration times for various protection and restoration methods considered (from [21]) ... 70

Table 5.2. Characterization of improved protection available with self-selected method (from [21]) ... 88

List of Figures

Figure 2.1. Typical real network topologies: (a) France (from [18]), (b) Italy (from [19]), (c) Europe (from [20]), and (d) USA (from [21]). 10

Figure 2.2. A diversely routed working and protection path pair in 1+1 APS (adapted from [27]). 13

Figure 2.3. Normal routing of the uni-directional half of a demand (a) and restoration routing (b) in an OPPR ring (adapted from [28]). 14

Figure 2.4. Demand flow in a span-restorable mesh network (a) and restoration action involving multiple restoration paths (b). 16

Figure 2.5. An example of SBPP primary and backup paths: (a) a normal working scenario, (b) a single failure and a successful backup path activation, and (c) a dual failure causing contention for spare capacity on a common span to both backups (adapted from [27]). 18

Figure 3.1. (a) An example *p*-cycle, (b) the protection reaction for an on-cycle failure, (c) the reaction for a straddling span failure (with two restoration paths), and (d) an unprotected failure on an "inside" span which is neither on-cycle nor straddling 28

Figure 4.1. Optimal solution to "Canada" network (a) illustrating that in the capacitated design, some *p*-cycles may be Hamiltonian (b)-(d) but others may not (e)-(f) (from [14] (p. 721), [70]) 53

Figure 4.2. Optimal solution to "20n40s1" network (a) again illustrating that a capacitated design may contain both Hamiltonian (b)-(c) and non-Hamiltonian (d)-(o) *p*-cycles 54

Figure 4.3. Single Hamiltonian (or two-cycle) solutions can use significantly more capacity than a standard multiple-cycle design (from [70]). 57

Figure 4.4. In a homogeneous, Hamiltonian network (a), a single *p*-cycle is optimal under both a hop-based (b) and distance-based (c) framework (adapted from [14] (p. 723),[70]). 59

Figure 4.5. Example to illustrate a non-Hamiltonian *p*-cycle (P1) having (a) a lower *AE* value, (b) an equal *AE* value, and (c) a higher *AE* value than a larger, Hamiltonian cycle (P2) (from [14] (p. 725),[70]). 62

Figure 4.6. Multiple-cycle designs required in non-Hamiltonian networks including those with three chains and two *p*-cycles (a)-(c), and one with five chains and three *p*-cycles (d) (adapted from [70]) (the graph in (a) is a variant of the Canada network seen earlier, while (b) is a variant of the well-known COST 239 graph as in [103]). 64

Figure 4.7. A semi-homogeneous network with two capacity levels supported by a Hamiltonian *p*-cycle (a), along with the implications for each type of span in both the whole-fiber (b) and wavelength switching cases (c) (from [14] (p. 726),[70]). 66

Figure 5.1. Test networks used: (a) COST 239 and (b) Generic U.S. (from [21]) 78

Figure 5.2. Illustration of different demand patterns specified for mixed service classes (in order of % APS - % *p*-cycles - % SBPP) ([107]) 79

Figure 5.3. Capacity results using assigned model with Generic U.S. network (from [21]). 81

Figure 5.4. Illustration of (a) a routing infeasibility that can prevent a second disjoint path from existing, and (b) a solution with two paths both avoiding the middle "rung" span. 82

Figure 5.5. Capacity results using assigned model with COST 239 network (from [21]). 83

Figure 5.6. Capacity results using self-selected model with Generic U.S. network (from [21]). 85

Figure 5.7. Capacity results using self-selected model with COST 239 network (from [21]). 85

Figure 5.8. Demand upgrading under self-selecting method design model in Generic U.S. network (from [21]) 86

Figure 5.9. Demand upgrading under self-selecting method design model in COST 239 network (from [21]). 86

Figure 5.10. For a single incremental working path (in red), a 1+1 APS solution (a) uses less capacity than the solution with a *p*-cycle (b). 87

Figure 5.11. "Square" network. 93

Figure 6.1. Example showing how large *p*-cycles with a path length (hop) limit can be more effective than imposing a simple circumference limit (from [110]). 98

Figure 6.2. Illustration of left (L) and right (R) path definitions used in subsequent ILP model, for (a) on-cycle spans and (b) straddling spans (from [110]) 100

Figure 6.3. Test networks used: (a) 13n23s, (b) 15n26s1, (c) 12n19s, (d) NSFNET, and (e) 19n35s (adapted from [110]). 106

Figure 6.4. Total capacity cost as hop limit increases, showing a threshold hop-limit effect (with designs from both mesh and path-limited *p*-cycle models) (adapted from [110]) 108

Figure 6.5. Hop-limited path distribution for 13n23s design ($C=7$, $H=6$) (from [110]) 110

Figure 6.6. Number of spare channels required as hop limit increases, again showing a threshold hop-limit effect with mesh and both circumference-limited and path-limited *p*-cycle designs (for network 19n35s) (adapted from [110]). 111

Figure 6.7. Hop-limited path distribution for non-joint, non-modular 19n35s design ($C=7, H=6$) (from [110]). 112

Figure 6.8. Effect of bi-criteria objective in improving path-length-limited results for 19n35s design ($C=7, H=6$) (includes previous results from Figure 6.7). 113

Figure 7.1. Test network used to examine variance effects (40n80s1-60s). 119

Figure 7.2. Results for variance check with 2,022 cycles in 40n80s1-60s 120

Figure 7.3. Spare capacity results for most methods over varying cycle set sizes (on 40n80s1-60s) 121

Figure 7.4. Test networks used in addition to those in Chapter 5: (a) 15n30s1, (b) 35n70s1-45s. 122

Figure 7.5. Performance (cost premium over optimal) for various network graphs. 122

Figure 7.6. Performance over varying nodal degree in 20n40s1 network suite. 124

Figure 7.7. Cycle set distributions (top to bottom: COST 239, Generic U.S., 15n30s1, 35n70s1-45s). 125

List of Abbreviations

λ	Wavelength
ADM	Add-Drop Multiplexer
AE	*A Priori* Efficiency (*p*-Cycle metric)
AMPL	A Mathematical Programming Language
APS	Automatic Protection Switching
BLSR	Bi-directional Line-Switched Ring
CIDA	Capacitated Iterative Design Algorithm (as in [18])
CoP	Class of Protection
DCPC	Distributed Cycle Pre-Configuration Protocol
DCS	Digital Cross-connect System
DP	Diverse Protection
DPCD	Distributed Preconfigured Cycle Design protocol (precursor to DCPC)
DRCN	Design of Reliable Communication Networks (workshop)
DWDM	Dense Wavelength Division Multiplexing
GA	Genetic Algorithm
HCC	Hamiltonian Cycle Cover (as in [68])
HCP	Hamiltonian Cycle Protection (as in [68])
HNC	Hamiltonian Cycle Neighbor Capacity Closure (as in [68])
ICC	International Conference on Communications
ILP	Integer Linear Program/Programming
IP	Internet Protocol (depending on context)
IP	Integer Program/Programming (depending on context)
ISP	Internet Service Provider
JCP	Joint Capacity Placement
L	Left (side) (as in Chapter 6)
LP	Linear Program/Programming
MIP	Mixed Integer Program/Programming
MMSCP	Multi-Method Spare Capacity Placement
MPLS	Multi-Protocol Label Switching
NP	Non-Polynomial
OADM	Optical Add-Drop Multiplexer

OCDC	Orientable Cycle Double Cover (as in [94])
O-D	Origin-Destination
OEO	Optical-Electrical-Optical
OFC	Optical Fiber Communication (conference)
OPPR	Optical Path Protection Ring
OSPR	Optical Shared Protection Ring
OXC	Optical Cross-Connect
PLJCP	Path Length Joint Capacity Placement
PSTN	Public Switched Telephone Network
PWCE	Protected Working Capacity Envelope
QoP	Quality of Protection
R	Right (side) (as in Chapter 6)
RPC	Restricted *p*-Cycle (as in [17],[69])
SBPP	Shared Backup Path Protection
SCP	Spare Capacity Placement
SLA	Straddling Link Algorithm (as in [18],[72])
SONET	Synchronous Optical Network
SSIU	Straddling Span Interface Unit
STS-1	Synchronous Transport Signal - Level 1
TS	Topological Score (*p*-Cycle metric)
TUM	Technical University of Munich
UPSR	Unidirectional Path-Switched Ring
VWP	Virtual Wavelength Path (i.e. with wavelength conversion)
WC	Wavelength Converter
WDM	Wavelength Division Multiplexing
WP	Wavelength Path (i.e. without wavelength conversion)

Chapter 1. Introduction

As information and communications technology becomes used in more and more spheres of human activity, and becomes even more ubiquitous and integrated into our everyday lives, there is attention being focused (quite naturally) on ensuring that these technologies and services are resilient. In virtually any engineered system, accidental failures are inevitable. Deliberate sabotage, while much rarer, is still possible in many cases. To ensure resilience, the systems we design must be able to cope with such failures, and ideally, do so in a way that completely insulates users from knowing that a failure even occurred.

1.1. Importance of Transport Networking

What may not be universally appreciated or apparent is the importance of the underlying transport networks that enable all the communication services we routinely enjoy. These networks are responsible for moving over distances the huge quantities of data aggregated from our collections of computers, cell phones, digital assistants, and more. They have incredible capacities and the actual level of aggregation and multiplexing might be surprising to many end users.

Historically, long-distance transport services were implemented through the use of terrestrial microwave or satellite transmission. Such methods had more limited capacity, but were the best available technology at the time. The later development of fiber-optic transmission media unlocked large quantities of new bandwidth, but also led to increased interest in survivability because of the greater concentration of traffic on one medium and the inherent vulnerability of fiber – it was not previously possible to "cut" the free space between microwave towers [1]. The resilience of optical transport networks is thus a crucial factor in their design, especially as more and more kilometers of fiber are deployed. If neglected, the consequences can potentially be severe.

In January 2003, a cargo ship navigating the Gulf of St. Lawrence decided to drop anchor (as related in [2]). Unfortunately the location where it did so coincided with the location of an underwater cable belonging to TELUS. Outage ensued until a backup link, under construction at the time, could be brought online four days later. The original cable was not repaired for almost two weeks. Ship anchors were also suspected in the damage of two much larger transcontinental cables, which carried a large portion of the Internet traffic from Asia to the United States [3]. As the article says, "Internet traffic in Asia was almost at a stand-still" [3]. Another transcontinental cable outage (with an unknown cause), this time in the Atlantic, was reported in [4], and here, because the cable was part of a survivable ring (more about rings later), the affected traffic should have been successfully rerouted. However, as [4] explains, a prior fault on the same ring had already occurred earlier in the month, and had not yet been repaired. This is a practical and very recent example of a dual-failure scenario that managed to affect the functionality of a network, even when single-failure survivability was assured.

Disasters on land can have a big impact too. Reference [5] describes a cable break in New York City that "cut off long-distance phone calls to much of the East Coast" and caused some Internet service providers (ISPs) to lose "up to 88 percent of their data during the early hours of the outage" [5]. In another incident, a train crash inside a tunnel was responsible for a major slowdown in the Internet as well. As a news report explained, "The subsequent fire severed cables and burnt through a massive Internet pipe serving seven of the biggest US Internet Service Providers" [6].

It is also worth noting that the failures just outlined are all quite recent (all within the last six years, and many more recent than that). While these anecdotes have been taken from news services, there are many articles on network survivability with a more academic flavor as well. These include [1],[7],[8] which approach the topic more from a legacy SONET/PSTN perspective along with more recent articles such as [9],[10],[11],[12].

With the importance of network survivability thus established more firmly with real-life examples, we can also motivate the continuing development of resilience methods by citing two other factors. The first, which is increasing in importance with the ubiquity of the Internet, is demand uncertainty. As [13] explains, "Networks are planned over a horizon of up to 5 years. It is now very difficult to predict exactly where the network will be in five years time in terms of traffic growth and distribution, technologies available to deploy, and demand for bandwidth." A very natural conclusion in the face of such uncertainty is that network design methods (including design for survivability) will need to transition to being even more robust and flexible than they presently are. Another reason why work on enhanced survivability is important is the economic and social aspect. The capacity growth that remains ahead can occur at a lower cost if more efficient protection and restoration methods are used, which can then result in lower costs for end users, hence accelerating the development of new network technologies and services, and in general, increasing the usefulness of the network to society.

1.2. Synopsis and Highlights

A recently discovered network protection method, called *p*-cycles, is attracting growing interest from the scientific, vendor, and telecom carrier communities. In this thesis, in addition to a thorough literature review, we present investigations into four timely areas of *p*-cycle theory and design that have recently come to our attention.

The first of these is in response to published work on Hamiltonian *p*-cycles and aims, in general, to show the limited applicability of this method, but while still examining it further from a research standpoint. We looked at the role of Hamiltonian *p*-cycles in a capacitated design and their cost compared to designs with a mixture of optimally selected large and small cycles. After looking at the *p*-cycle approach in environments with homogeneous and non-homogeneous capacity, we suggested a semi-homogeneous type of *p*-cycle network that exactly achieves a well-known lower bound for any type of span-restorable network. This work appears in Chapter 4, which is the first of the four research-oriented chapters in this thesis.

Inspired by the idea of ring-mesh hybrids in the literature, we then proceeded (in Chapter 5) to propose a new design framework for networks incorporating the 1+1 APS, *p*-cycle, and shared backup path protection techniques. We found that not only did a capacity savings result from this approach, but that a significant number of lightpaths were able to "upgrade" to a better class of protection than they strictly required. Even when all demands needed only the lowest grade of service, in one of our test networks fully 45.4% of them enjoyed protection with a better technique, at no increase in cost. We found that the upgrading effects were more pronounced in the sparser of our two test networks, and explained why this upgrading occurs (at a reduced overall network cost) in spite of the fact that these faster methods are generally thought to be more expensive.

We then examined the issue of path length constraints in a *p*-cycle framework, and this study is presented in Chapter 6. In prior design methods there has never been a way to strictly limit the length of *p*-cycle restoration paths, which could potentially make their way through a very long *p*-cycle while the network is in a failure state. In this section we give a model that can implement this path-length-limiting capability and we use it to compare designs with limits imposed on individual path lengths as well as overall cycle circumferences. Our main finding was that the computationally easier and more common method of simply limiting circumferences is an effective surrogate for rigorously limiting the lengths of the individual protection paths. We also used our model with path length restrictions to show that a threshold hop limit effect does exist in *p*-cycle networks, similar to the effect by the same name previously reported for span-restorable mesh networks in the literature.

While conducting these three investigations, we were reluctant to impose any preselection (i.e. filtering) mechanism to scale down the sets of candidate *p*-cycles for selection in the various designs. Doing so could introduce an obfuscating factor because the present methods are known to sometimes produce an inferior candidate set, which will then lead to non-negligible differences in the optimization results. Our final study in Chapter 7 is therefore a brief look at a new idea for preselection based on matching the statistical distribution of cycles in the filtered set to the distribution of the full population of all cycles. We expected this to be a superior method of *p*-cycle preselection, and in general the results were quite promising. However, the new method also produced highly variable results, which might limit its usefulness as a practical strategy.

Together, these four topics are an interesting supplement to the current body of knowledge on *p*-cycle techniques, but another contribution of the thesis is a scholarly one of presenting the first comprehensive review of all known *p*-cycle literature. This appears in Chapter 3, prior to the four research chapters just outlined, and is the fifth significant contribution of this thesis. The literature review covers the historical development of the *p*-cycle method, work on WDM considerations and wavelength conversion effects, Hamiltonian *p*-cycles, preselection and heuristic methods, *p*-cycles used with ring mining, flow *p*-cycles and path segment protection, dual-failure survivability, and miscellaneous other *p*-cycle ideas proposed in different papers over the years. References to applicable overview papers and patents are also included, and a suggested reading list of the "best" *p*-cycle papers (for a reader new to the field) is presented at the end.

Before we move into these five central chapters and the subsequent summary, concluding remarks, and suggestions for future work appearing in Chapter 8, we begin now in Chapter 2 by stepping through a background discussion of transport networking, including optical networks, terminology, metrics, protection and restoration strategies, and a brief explanation of integer linear programming (ILP) methods as applied to survivable network design.

Chapter 2. Background Concepts

To start with, we will cover a general overview of optical transport networks, followed by a discussion of some common protection and restoration techniques, including a more in-depth summary of methods we use later on (with the notable exception of *p*-cycles – due to its importance, an introduction to the *p*-cycle technique appears separately in the next chapter). A brief review of optimization methods is also presented due to their prominence in obtaining our later results.

2.1. Optical Transport Networks

2.1.1. Technologies and Basic Definitions

A transport network resides immediately above the physical equipment in a layered, hierarchical view and provides transport functionality to logical networks and services at higher layers. It is concerned primarily with ensuring that bulk data transmission is fast, efficient, and survivable. It is not concerned with the exact nature of the data services it carries, nor with their initial origins or final destinations outside the transport network. The concept of a transport network is generally not difficult, but there is a specific set of terminology, which we will briefly step through next. This overview is based in part on the information in Chapters 1 and 3 of [14] (pp. 15-60, 103-172), which contain many further details not covered here.

A transport network is constructed in a physical layer graph, and is comprised of nodes and spans[1]. A node is simply a location where demand units can be added or dropped from the network. A span is the collection of all transmission capacity between a pair of nodes. Even if the capacity between two nodes is on different physical fibers or has physically diverse routing, it is still considered to be one span. Each span is made up of a number of channels, which are single units of capacity at the granularity of the particular system in use. For this work we will typically assume that a channel is a single wavelength in a WDM fiber-optic system but there are many other possibilities, including

---

[1] The use of the terms "span," "link," and "channel" is not, in general, consistent throughout the literature. In this thesis we use the word "span" to represent the entire group of all "links" or "channels" (equivalent for our purposes and used interchangeably) between node pairs (see [15] (p. 118)).

STS-1 links in a legacy SONET system. Although the concepts and details of the work presented are more general in nature and are not intrinsically specific to any one technology choice, for convenience we will use WDM terminology throughout.

The term "optical network" can have a variety of meanings. In the most general case, any network using fiber-optic media is an optical network. In another (increasingly used) context though, a "WDM" optical network is one where the individual channels are wavelengths. In such a case, multiple wavelengths (λs) are carried on a span using WDM, or more typically, DWDM technology. A SONET network using a single wavelength on each fiber to carry the SONET payload is not really considered an "optical" network in this framework, even though its transmission takes place over optical fiber. A convenient way to check is to "look inside" a cross-connect, ADM or other nodal device. If the channels being manipulated inside the nodal element are wavelengths, then the system is an optical network in the current sense used to refer to WDM or DWDM networks.

2.1.2. Network Demands

Nodes in a transport network exchange demand units. These are comprised of traffic flows from higher layers that are groomed and multiplexed into the appropriate format and then injected into the transport network. It is completely irrelevant to the transport network what particular services are being groomed into transport demands. Common payloads include Internet (inter-router) traffic, voice traffic, and traffic on leased lines that make up internal government or corporate data networks. Grooming simply refers to the process of neatly "packaging" component traffic flows into transport network demand units, together with other traffic flows sharing an identical or nearby destination. This increases efficiency (similar to how airline passengers are groomed to fly between major "hub" airports instead of directly from their origin to final destination).

It is not necessary for a particular transport network to have a node at every location that requires communication services. Transport networks exist at different scales, each providing service to a particular set of nodes over a particular geographic region. A metro network will typically exist within a single city, while a long-haul network could span an entire continent. These networks can interact and carry demand units for each other – what one network sees as a single node may in fact expand (at a lower level) into an entire metropolitan or regional network to serve a given city and its outlying communities. Just because a city or town is not a node on a given transport network does not mean that the (typically large) network will not serve that location – it may do so indirectly through a smaller regional network. (This can be seen in Figure 2.1 later in this chapter. Note that two of the networks pictured are for individual European countries while another is for Europe as a whole.)

For the purpose of this work, demand quantities are assumed to be bi-directional. This is a common assumption in other work done in survivable networking (if "transmission capacity is implicitly bi-directional" as stated in [14] (p. 176), then logically we can see that demand units must be as well). This approach is actually conservative because in the case of a real-life demand imbalance, the assumption of bi-directionality will over-estimate one of the two demand quantities and thus cause the network to be over-provisioned (admittedly somewhat wasteful but easier to design and provision, and also always guaranteed to be adequate for the imposed demand). Any network topology graphs and demand patterns to follow are thus undirected (equivalently, they are implicitly bi-directional), although for ease of description we will describe a demand as having an origin and a destination (an O-D pair), which can be interchanged with no effect. A helpful analogy is to think of a telephone call. It is a bi-directional data stream but can be represented as having an origin (the caller) and a destination (the called party).

A demand unit typically corresponds to the capacity of a single channel. Thus, a unit demand requires a continuous transmission path comprised of single channels that connect the origin and destination. The most obvious objective of a transport network with respect to these demand units is to satisfy them, that is, to successfully provide a transmission path for every demand unit from its origin to its destination.

2.1.3. Failures and Survivability

Another objective of a transport network is to provide some form of survivability so that some (or more typically, all) demands will be protected in the event of failure. As DWDM technology improves and even more channels are carried on a single piece of glass, the importance of this ability to provide protection cannot be underrated. In this work (like much other work in this field), we assume that a span failure is complete (when a span fails all of its individual channels fail). In many cases this will certainly be true, but in other cases, certain channels may survive. Our assumption of total failure thus may not always be perfectly accurate, but is definitely conservative in the sense that we are always looking at the worst-case scenario, and it is the case we must plan for.

We classify capacity in a network into two categories: working and spare. Working capacity is used for routing of actual demand units. Spare capacity is used in the event of a span failure to provide restoration paths for the affected working demand units. (More generally it can also be used in the event of a node failure to protect the transiting flows passing through the node that failed.) There is also typically a modularity overhead involved as commercial systems are sold only in certain sizes. The spans in a network may thus have excess capacity beyond what is strictly required for working and spare allocations. Such excess capacity could also be provisioned intentionally as a cushion to cope with dynamic demand (as in the PWCE concept [16]).

2.1.4. Nodal Degree, Redundancy, and Typical Networks

Having considered the basic structure and objectives of a transport network, we can now look at a few metrics and terms used when considering the network graph from a more analytical viewpoint. If we examine each node of the network, we can find its nodal degree (d) by simply counting the number of incident spans. Note that in a survivable network, the absolute minimum degree for any node is 2 – if a node were connected to the rest of the network with only a single span, the failure of that span would cause the node to be completely disconnected from the rest of the network regardless of the survivability mechanism that was in place. Normally we do not pay much attention to the degree of individual nodes, however, and are usually more interested in the average nodal degree of the network ($\overline{d}$). This gives a general indication of the sparseness or richness of the graph. Typical real transport networks range from just above 2.0 (on the low end) to somewhat below 5.0 (on the high end). In Figure 2.1 we illustrate some real networks taken from the literature to show the range of average nodal degrees and what typical topologies might look like. Due to geography, demography, and history, many North American networks are quite sparse, while networks in Europe, on average, are more richly connected.

After seeing these examples, we can quickly make the point, in response to some other work in the networking field ([17], for example), that the use of fully meshed network models is largely inappropriate in making comparisons between survivability techniques. (A fully meshed network is one with a span between every node pair, and thus obviously has $\bar{d} = N - 1$, where N is the number of nodes.) It is fairly easy to appreciate that these are graphs of the physical layer facility routes: for any more than, say, four or five nodes, no telecommunications carrier would attempt to build separate physical spans between every single node pair. For a North American operator, even increasing the average nodal degree up to around 3 or 4 may be prohibitively costly.

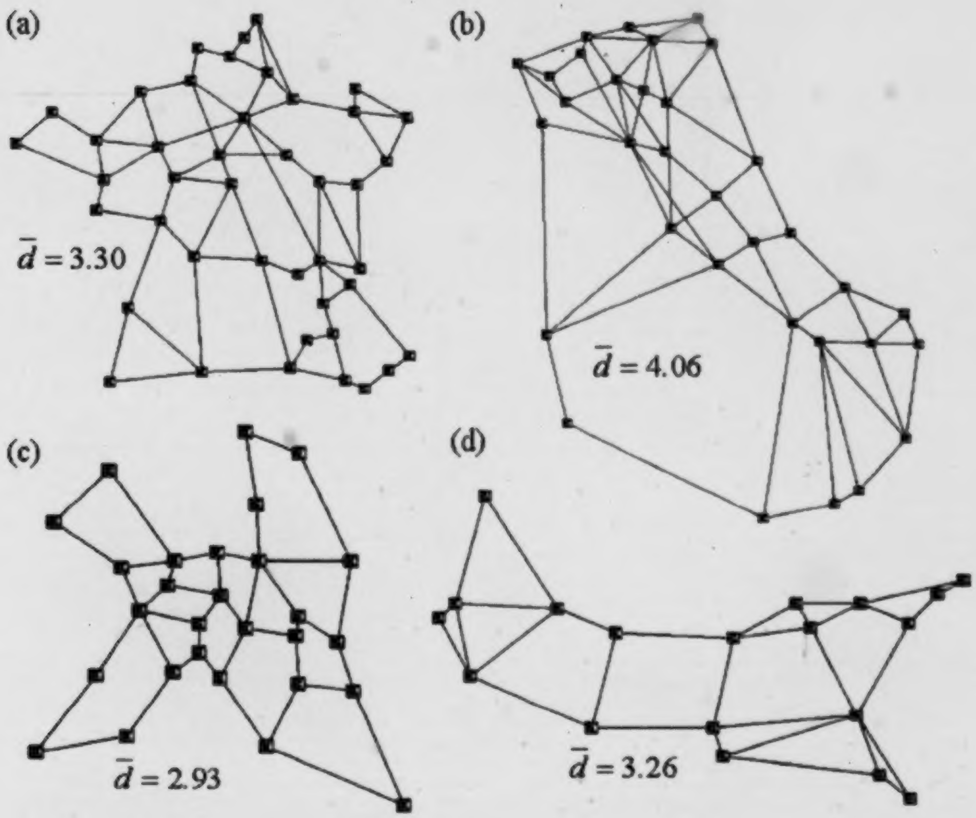

Figure 2.1. Typical real network topologies: (a) France (from [18]), (b) Italy (from [19]), (c) Europe (from [20]), and (d) USA (from [21]).

To ensure restorability, individual nodal degrees of at least 2 are necessary, but not sufficient, however. Depending on the types of failures we consider, the network must be either "two-connected" or "bi-connected" for a restoration mechanism to be effective in all cases. As explained in [14] (p. 180), two-connectedness means that at least two *span-disjoint* routes exist between every node pair, while bi-connectedness requires at least two *node-disjoint* routes. A bi-connected network (with enough spare capacity and an appropriate restoration mechanism, of course) is thus prepared to cope with any span or node failure that could occur.

Once we have a bi-connected topology, we might ask about the cost of provisioning enough capacity to actually make it restorable. This is of course dependent on the survivability method we choose, but all schemes can be evaluated with the common metric of redundancy, which is defined in general as the ratio of spare to working capacity present on all spans. More formally we can define redundancy as

$$\Re \equiv \sum_{\forall j \in S} c_j \cdot s_j \Big/ \sum_{\forall j \in S} c_j \cdot w_j , \tag{2.1}$$

where S is the set of spans (indexed by j), c_j is the length of span j (or in general, any other measurement of cost), and w_j and s_j are the number of working and spare channels on span j, respectively. In a hop-count framework where all spans are assumed to have equivalent cost, we would simply set all values of c_j to be 1. In many cases, though, it is better to measure the distance-weighted redundancy, as this will give a more accurate picture of the total network cost, especially for long-haul networks where fiber costs may be more significant than the cost of nodal equipment.

2.2. Protection and Restoration Techniques

As we have seen already, optical transport networks and the demand units they carry are extremely important, simply because of the high levels of aggregation and variety of applications being carried on a single lightpath. Here we review some common methods for protection and restoration in a transport network. Although the terms "protection" and "restoration" may seem similar in that both involve the recovery and rerouting of demand units traversing a failed span, there is actually an important distinction. A protection method is one where the action to take upon failure is known in advance and the recovery lightpaths are all preconnected (and ready to go). On the other hand, a restoration method is one where either the recovery actions and switching need to be determined on the fly, or the lightpaths need to be connected on the fly, or both. For example, a preplanned span-restoration mechanism (as in [22]) may know, for every failure span, what individual channel-level protection paths need to be created to restore the traffic, but cannot connect them in advance because the pathsets for each failure are all different (and, of course, it would not know exactly which span will fail next). Another possibility is a fully self-organizing response [23],[24] that not only needs to connect the recovery lightpaths post-failure, but will not even know what lightpaths are possible until it discovers them. Thus, while both protection and restoration methods can be designed to effect a full recovery, a protection method will typically do this by simple, autonomous switching at a pair of end nodes, where a restoration-based scheme will have elements of path discovery and/or connection to contend with in real-time, post-failure.

Each method may have different average requirements for the total spare capacity needed to achieve full restoration, and each may offer a different typical recovery time. As a very general rule, those methods offering very fast recovery will need larger amounts of spare capacity than those methods which are relatively slower. (It will be seen, however, that *p*-cycles are a notable exception to this rule.)

2.2.1. Automatic Protection Switching

Automatic Protection Switching is a very simple and historically common technique for designing survivability into a network. More complete descriptions are given in [14] (p. 119),[25], but for our purposes here, we can simplify the description to say that this method enables survivability by making use of a fully-connected backup path to restore traffic in the event of failure. We use a specific variant called 1+1 APS, where a completely separate protection path is provisioned along with each working path (a one-to-one correspondence). The signal is sent simultaneously along both paths and the receiving node can switch between them in the event of failure on the working arm. Although this is the general definition, the discussion later in Chapter 5 will use the 1+1

APS terminology to refer to a fully disjoint special case called 1+1 DP. As [14] (p. 119) explains, "1+1 DP implies that the protection channel is routed over physically diverse rights-of-way from the working system." This essentially means that, barring a freak coincidence, both protection paths in a 1+1 setup will not suffer an outage at the same time. This is what we are looking for when the APS method is used to protect entire spans.

Because 1+1 APS has a totally dedicated protection channel already carrying the signal, it is typically used for premium services and those requiring high availabilities. There will never be any contention for a protection channel, and so protection switching can be extremely fast, approaching or reaching the well-known 50 ms restoration time standard. (More recently, though, the necessity of this requirement has been challenged [26].) However, the disadvantage of APS (which may already be apparent) is its wastefulness in the capacity realm – unlike many other survivability methods, there is *no* sharing of capacity between different failure scenarios *at all*. Fully duplicating every working path in a network can be extremely expensive, implying a 100% redundancy in the best case. Normally, though, the protection path, in order to be disjoint, will need to take a longer route by either hop counts or distance, making the typical redundancy even higher. An advantage of 1+1 APS design, however, is its utmost simplicity because there is no element of optimizing any kind of tradeoff – a simple algorithm for finding disjoint path pairs (or the shortest cycle) will perform quite well. In Figure 2.2 a typical 1+1 APS working and protection path pair is shown for further clarity.

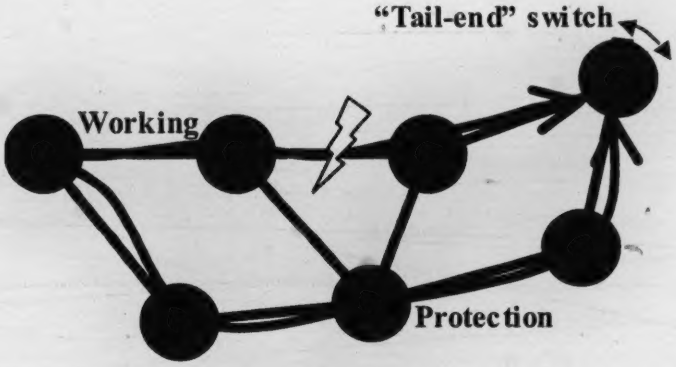

Figure 2.2. A diversely routed working and protection path pair in 1+1 APS (adapted from [27]).

2.2.2. Ring Protection

In the 1990s, industry attention was largely focused on ring networking. Although the idea of mesh-survivable network design was appearing in the literature (e.g. [22]), the mesh-networking paradigm did not then meet with widespread acceptance. In a ring network, working and spare capacity are arranged in cycles (rings) and are coupled (each ring must have equal amounts of working and spare capacity). The spare capacity can only be used to recover from span failures affecting that particular ring. When a span on the ring is cut, the affected demands are simply routed over the spare capacity on the surviving spans. An example of this is shown in Figure 2.3.

Here we can see the operation of an optical path protection ring (OPPR), where we note that the bi-directional demand is split and routed in both directions around the ring on the two uni-directional working fibers, as explained in [28]. (Figure 2.3 only shows a single, uni-directional half.) Because each span will thus carry all traffic associated with the ring, OPPR is very much like 1+1 APS in that the total spare capacity required is the sum of capacity on all working paths requiring protection. (This type of ring is essentially a collection of logical 1+1 APS setups arranged in a circle as necessary between each demand's entry and exit points to and from the ring.) Another (more advanced) variant, called an optical shared protection ring (OSPR), keeps both directions of the demand flow together, which therefore allows the working capacity on spans of the ring to vary, as individual demands are routed only over a single side. Redundancy levels

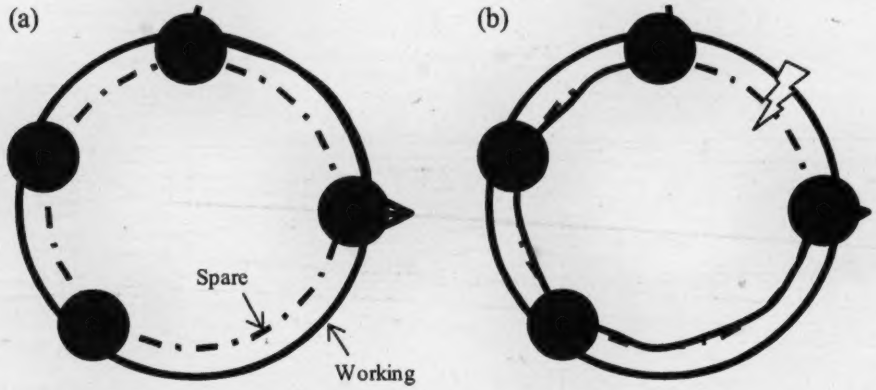

Figure 2.3. Normal routing of the uni-directional half of a demand (a) and restoration routing (b) in an OPPR ring (adapted from [28]).

are typically much lower in this case than for the less efficient OPPR type (more details can be found in [28]). (These OPPR and OSPR ring types are logically equivalent to the UPSR and BLSR rings in the SONET framework, respectively.)

Because the protection path is known in advance (inherent in the fixed structure of the ring), the only real-time action required is the actual switching itself, which makes ring networks very fast, similar to those protected with 1+1 APS. (This was one factor responsible for the initial popularity of rings as an APS alternative.) However, the disadvantages of rings are numerous. Redundancy levels are 100% at best. In typical ring networks, once stranded or unused working capacity is accounted for, redundancies (relative to used working capacity) can go to 200-300% [29]. In addition, the problem of optimally designing networks comprised of multiple rings is "notoriously difficult" [30]. By their nature, rings are also less flexible because of the fixed relationship between working and spare capacity. These issues are making ring networking a less attractive option over the longer term as demand quantities continue to grow (often unpredictably).

2.2.3. Mesh Restoration

The late 1990s to the present have seen the gradual emergence of mesh networking into popular favor. Mesh networks use an OXC (optical cross-connect) as a nodal element and treat the wavelengths as a sea of channels that can be used as either working or spare. This allows much more flexibility in the routing of both working and protection paths (which can now be completely unassociated with each other). A common practice is to use a simple shortest path algorithm to route the working paths and to then solve an integer linear programming (ILP) model to determine the placement of spare capacity and the routing of protection paths. This method is known as spare capacity placement (SCP) because the optimization engine is only solving for the spare capacities. An alternative method optimizes the routing of both working and protection paths together and is consequently known as joint capacity placement (JCP).

Note that in a mesh network, the demand flows that would normally traverse a certain failed span do not need to remain together during restoration. One of the advantages of dynamic mesh restoration is the ability to use a number of distinct paths for routing the failed working demands, and these do not have to remain "near" the failure, but can take

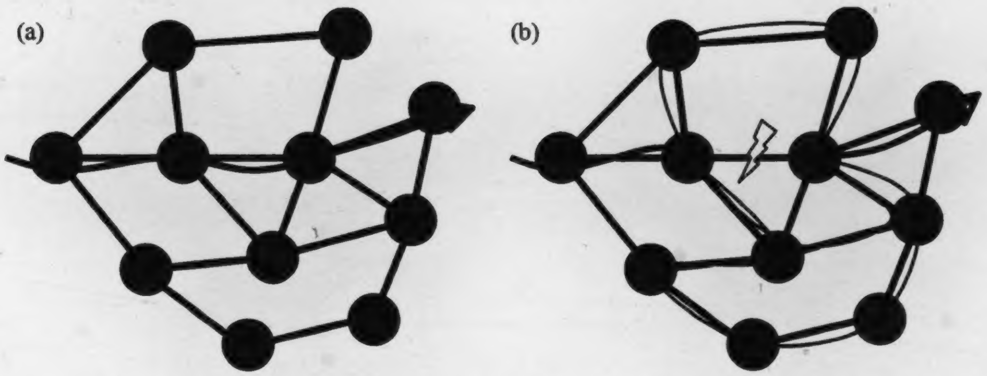

Figure 2.4. **Demand flow in a span-restorable mesh network (a) and restoration action involving multiple restoration paths (b).**

any reasonable route through the network, as long as this contributes to an efficient global solution. Mesh restoration can be sub-categorized into two main variants. Span restoration finds restoration paths for affected demand units between the end-nodes of the failure span. Path restoration, on the other hand, finds restoration paths between the origin and destination nodes of each demand affected by the failure. Studies have shown that path restoration is very efficient in terms of capacity [31], but it relies on a more complex real-time mechanism that makes it slower than span restoration. Figure 2.4 shows a typical span-restoration path-set to illustrate how the total demand quantity can be separated into unit demand flows which traverse a number of different (but not disjoint) routes.

2.2.4. Ring-Mesh Hybrids

Both the ring and mesh options are certainly viable when used alone, but they can also be used together as the two elements in a coordinated strategy. To explain these "ring-mesh hybrids" we first need to understand what a forcer is [32],[33] in a mesh-restorable network. As [33] explains, "*a forcer is any span for which an increase in network total sparing is required to maintain restorability if the span's working capacity is increased.*" Basically, this means that such a span will "force" the spare capacity level to be what it is. Not all spans in a network will necessarily be forcers, making it possible for non-forcers to support some additional working capacity at no extra cost (up to and including the point where the span being considered becomes a forcer itself, beyond which extra spare capacity will obviously be required).

Work to date on ring-mesh hybrid design is summarized in [33]. The approach taken in that paper was to see if the strategic placement of rings to protect the largest forcer spans could reduce the cost of the remaining mesh portion in the network by a greater amount than the cost incurred in building the rings. Each survivability method would be independent in the sense that demand units would only be assigned to one of the two methods and each method would implement its own protection or restoration mechanism independently of the other. The only relationship would thus be in the design model itself when deciding which demands should be protected by which method, and where the actual rings and the mesh spare capacity should be placed.

As the concluding discussion in [33] points out, the idea of using rings strategically to reduce the effects of forcers can "significantly reduce the cost of a mesh network by incorporating rings chosen for forcer-clipping effects." However, it also notes that even as the cost of rings is made progressively less costly in the design model, the solutions returned were still hybrids. The authors state, "We think the understanding of these architectural interactions is sufficient to say that in general a hybrid should be lower in cost than either a pure mesh or pure ring network over a fairly wide range of relative equipment costs" [33]. This fundamental advantage of a hybrid design approach is something we will return to and use as a motivating point in Chapter 5 of this thesis.

2.2.5. Shared Backup Path Protection

A popular method at the present time is Shared Backup Path Protection (SBPP), sometimes called Shared Mesh. This is a restoration scheme where each "primary" (i.e. working) lightpath is assigned a single backup path at the time when the primary path is established [34]. In that sense it is somewhat similar to 1+1 APS. However, the aspect of sharing comes from the ability of multiple backup paths to rely on the same spare capacity, when their corresponding primary paths are fully disjoint. Because these primary paths have no relation to each other, any single failure that could occur would only disconnect (at most) one of them, and thus there would only be one primary path that would require a backup path to be formed for restoration. This is illustrated in

Figure 2.5, where in (a) we can see two primary paths and the preplanned routes for their corresponding backups. Note that the backup paths have certain spans in common, and on these spans we would only provision enough spare capacity to accommodate one backup at a time. (Backup paths in SBPP are not preconnected so prior to failure they will only exist as a route description and not a physically cross-connected sequence of channels.) In part (b) a single failure and activation of the appropriate backup path is shown. A dual-failure case is shown in part (c) and in this case the mechanism cannot restore both because there is contention for the shared capacity. Sometimes a limit is imposed on the extent of capacity sharing in order to mitigate the effects of dual failures.

SBPP is similar to span-restorable mesh restoration in that the backup paths need to be cross-connected on the fly, but quite different because only one backup path (determined in advance) for each primary path is possible. Span-restorable mesh

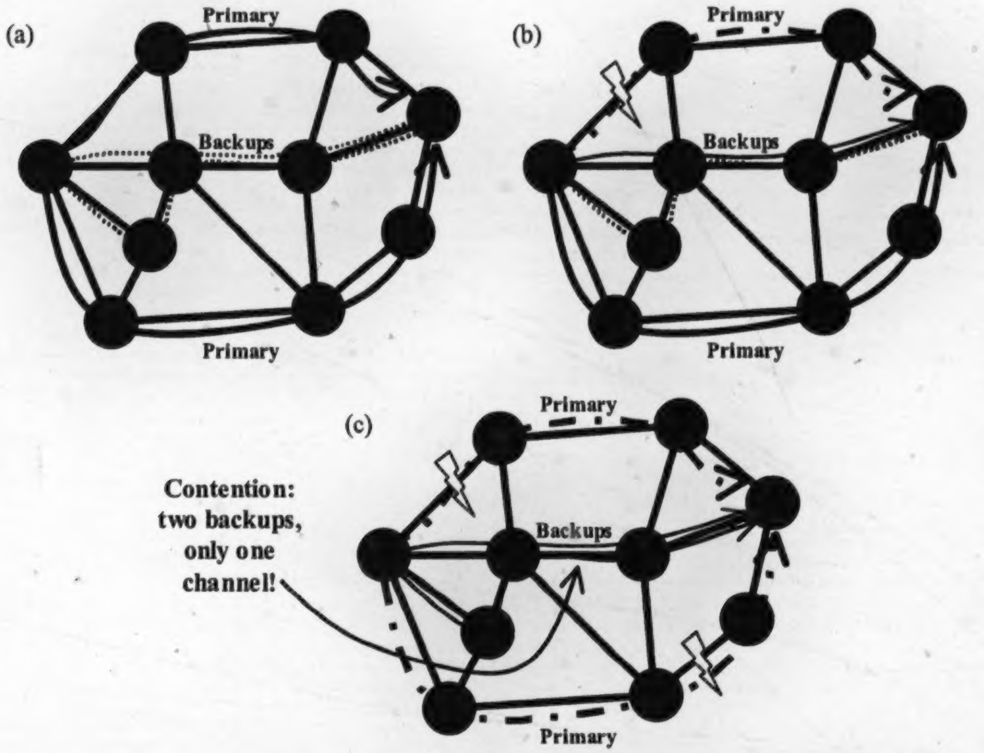

Figure 2.5. **An example of SBPP primary and backup paths: (a) a normal working scenario, (b) a single failure and a successful backup path activation, and (c) a dual failure causing contention for spare capacity on a common span to both backups (adapted from [27]).**

restoration thus has the advantage of being able to adapt more readily to events such as dual failures. The SBPP method is also at a disadvantage compared to *p*-cycles because of the need to connect restoration paths in real-time (*p*-cycles are preconnected). The main advantages of SBPP are the low redundancy levels that can be achieved and simplified failure detection because a failure anywhere on the path triggers the same end-node switching reactions. Fault isolation is not needed. The sharing of spare capacity over a number of backup paths makes it a very efficient method. Reference [35], for example, shows the somewhat better capacity performance of SBPP compared to many other techniques.

2.3. Optimization and Integer Linear Programming

As [36] (p. 137) explains, "Linear programming (LP) techniques are widely used to solve a number of military, economic, industrial, and societal problems." Among these applications are optical network design studies, and we make use of optimization techniques as the primary tool in obtaining the results and conclusions in this thesis.

2.3.1. General Background

First, it is important to understand that "programming" in this context is not the same as we would use it conventionally today in reference to programming a computer. Instead, the term was intended to indicate a plan or a schedule of some sort, along the lines of the "program" at a conference or event, for example. The program for a conference is a good example of a problem we might use optimization techniques to solve. Imagine a large conference with a number of speakers with varying availabilities, similar papers where presentations should be grouped together, limited facilities, a desire to keep the event as short as possible, and a need to determine a detailed conference schedule. A quick and simple algorithm for finding the best answer is not easily developed, but an optimization engine would handle a problem of this nature quite easily. In our description above, we implicitly included the five elements to a "program" (more typically called a formulation or a model), which we go over next.

2.3.2. Problem Formulation

To solve a particular problem with Integer Linear Programming (ILP), we need to first express the problem in a standard way. Each model (or formulation) has five parts: sets, parameters, variables, an objective function, and constraints.

SETS

A set is a collection of elements to be considered somehow in the optimization. In our conference planning example, we might need to consider a set of rooms or a set of speakers. The elements in a set are typically denoted only by name and do not have detailed information encoded with them as part of the set.

PARAMETERS

Parameters are input data that is known in advance. This is where detailed information can be introduced to the problem. For example, parameters might be the total attendance limit or the maximum number of sessions per day. These parameters are more general, but we can also specify parameters that are indexed over the members of one or more sets. Examples of these could include the seating capacity of all rooms (a set), or the availabilities of all speakers (also a set).

VARIABLES

Variables are what we are trying to find optimal values for, and the optimization engine will assign values to each variable we include. They can again be general (such as the total cost of renting facilities) or specific to the elements of one or more sets (for example, the expected attendance in each room).

OBJECTIVE FUNCTION

While in some cases we only need to find a single, workable ("feasible") solution, most often we can identify some criteria for choosing between alternate solutions if possible. For example, at a conference we might try to minimize the program duration or the cost of facilities. In general, the objective function has both a "sense" (minimizing or maximizing), and a function to calculate the exact value. It is also possible to have what is called a "bi-criteria" objective where two different quantities are included, with the second of these weighted by a conversion factor expressing its value in units of the first.

CONSTRAINTS

To avoid trivial solutions, like scheduling all speakers in parallel and ending the conference on the morning of the very first day, we also need to express constraints that specify the properties that an acceptable solution must have. For example, we could say the number of parallel sessions must be no greater than three. In addition to the constraints we specify to represent each problem, an additional implied constraint in every ILP model (sometimes explicitly indicated and sometimes not) is that all variables must be non-negative. Occasionally variables will be restricted in other ways too, as we will see in the next section.

2.3.3. Linear vs. Integer Programming

Although the problems are formulated in the same way, as outlined above, there is an important difference between those that are "linear" and those that are "integer." Both of these actually impose the requirement that all functions of the variables must be linear. But in what we usually describe as simply a "linear" program, other than the restriction against negative values, all variables are unrestricted and can take on any real value, including fractions. For example, the total cost of renting rooms for an event would be a real variable (it would not need to be an even dollar amount). On the other hand, for an integer program, all variables must be integer, that is, they are restricted to taking on integer values. An example of this might be the number of rooms required (renting a fraction of a room makes no sense). But again, the *function* of the variables in an integer program must still be a linear relationship.

A large number of programs will have a combination of these types of variables – some real and some integer. In this case we call the formulation a Mixed Integer Program (or MIP). It could also be called an Integer Linear Program (or ILP), and this is also the terminology we use for the general case when we are not quite sure how to classify it. Another common designation applied to IPs, MIPs, and ILPs is the 1/0 prefix. This means that at least one of the variables, in addition to being integer, is further restricted to being only 1 or 0 (or any other pair of binary choices). An example of a variable like this might be whether or not a certain room is used on a certain day – the answer can only be "yes" or "no." A further subset of problems is comprised of those that are "pure 1/0" – this means that *all* the variables are binary. In general, problems with a large number of 1/0 variables are more difficult to solve than other types.

Commercially available ILP solvers make use of a number of mathematical techniques, the basics of which are explained in optimization textbooks such as [36],[37], and in many other sources.

2.3.4. Practical and Scientific Benefits of ILPs

A common attitude expressed in the literature on network design is that because ILP problems are NP-hard (a designation that indicates exponentially increasing solution time as the problem size grows), they are not an appropriate technique or solution method and thus not worth pursuing. Theoretically we can appreciate that any problem classified as NP-hard becomes practically unsolvable at some point. It is crucial, though, to find out exactly what point that is, and make use of ILP methods to the considerable extent that we can in practice. The argument that ILPs are not inherently scalable and cannot solve large problems is a bit like arguing that a pocket calculator is not scalable – technically true, but if the method is still useful for our purposes, why not employ it? Problems on a network of 1000 nodes could perhaps take millions of years to solve, but problems of practical interest on real networks with only a few dozen nodes often solve in hours, minutes, or even seconds.

As engineers, we can thus appreciate the usefulness of ILP techniques in finding efficient designs for practical networks. And as researchers, we can consider ILP methods as a kind of microscope that lets us examine what is fundamentally true about various network architectures (as [14] (p. 224) points out). In these cases, except for the obvious requirement that we be able to solve the problems in a reasonable time (and not the life of the universe), the actual run times become irrelevant because we are looking for insight into the ideas we have chosen to study – how long we spend "looking through the microscope" is not that important. In all the results presented in later chapters, run times are not included for this very reason.

2.3.5. ILPs for Network Design

In designing survivable optical transport networks, we frequently want to minimize the cost of the design solutions and find an arrangement of working and/or spare capacity that will allow all working demands to be successfully routed while allowing enough spare capacity, arranged as necessary, for all demands to be restorable in the event of any single span failure. Typically, elements such as spans, routes, cycles, capacity modules, etc. are represented as input sets. Parameters are also specified to give the properties of these elements, such as whether a cycle crosses a given span, or whether a particular route can be used to restore demand units in a certain failure case. The solutions are encoded in variables representing quantities such as the flow over a restoration route, the spare capacity on a span, or the number of copies of a given *p*-cycle. As mentioned, our objective is frequently to minimize some variant of cost (spare, total, or modular), and the constraints are specified based on the exact survivability method being used. Examples of these might be constraints to ensure that all working demands can be restored or constraints to ensure enough spare capacity is supplied to accommodate the flows on all routes.

Frequently, however, specifying the complete sets of input elements can overwhelm the optimization engine and prevent a solution from being found in any reasonable amount of time. In one important context, that of designing span-restorable mesh networks, this impasse in using ILP methods to design networks was greatly ameliorated when Herzberg et al. proposed that the input sets simply be limited to a more restricted (and much more manageable) set of elements [22],[38]. Obviously we cannot remove any critical items such as the individual network spans, but instead of representing every possible route through the network graph, we could perhaps scale this back to only 5, 10, or 20 routes between each of the O-D node pairs. This is what Herzberg and colleagues proposed in the two papers just cited. In this case we can still use the same theoretically accurate design models, but simply limit the inputs we give to the optimization engine, to avoid overwhelming it. An important finding reported in Herzberg's papers above is that a threshold effect was observed, where allowing progressively more and more routing options in the input data sets produced literally no further improvement in the solutions, above the threshold hop limit. The absence of any improvement above the threshold indicates that the strategy of limiting the input sets can lead to the same quality of solutions as with the full data sets, at least for the span-restorable mesh technique Herzberg was considering.

We employ this technique in some of our solutions in a later chapter. To obtain these solutions, we used commercial optimization software called ILOG AMPL and ILOG CPLEX. AMPL is both the name of a program and a modeling language – optimization models can be specified in the AMPL language and provided as input to the AMPL software, coupled with a data file containing the contents of the input data sets that were specified in a generic, abstract way as part of the model. AMPL parses and combines these two files into an overall specification of the problem in the format required by CPLEX. The CPLEX optimization engine then engages in the process of using various algorithms and optimization techniques to find the best solution to the problem (i.e. a solution that minimizes or maximizes the value of the objective function as much as possible). A tolerance can be set so that when a solution adequately close to optimal is discovered, CPLEX will halt and return that solution, instead of engaging in a futile and lengthy search for a solution that might only be 0.01% better.

In the TRLabs Network Systems group, two CPU servers were available for running AMPL and CPLEX tasks. The machine named "apollo" was a Sun Fire V280 server with four 450 MHz processors and 4 GB RAM, running ILOG CPLEX 7.1, and a newer machine, "liza," was a Sun Fire V480 server with four 900 MHz UltraSparc III processors and 16 GB RAM running CPLEX 7.5.

The other tool we make use of is a suite of custom 'C' programs to analyze the network topologies and demand patterns in each specific instance to generate the AMPL data files that are required. The input set "filtering" described earlier, or "preselection" as we often call it, is implemented in these programs. Many of them (or at least their "building blocks") were readily available in the Network Systems group as part of a shared library of programs and code segments arising from previous research done by members of the group over time. The specific AMPL models and preprocessing programs we use are explained further in subsequent chapters.

Already in some of our discussion in this chapter, we have mentioned *p*-cycles in passing or used them as examples. An explanation of the *p*-cycle method has been conspicuously absent from this background discussion, but for good reason: there is enough literature and prior knowledge on *p*-cycles to warrant a fully separate and complete presentation, which we look at in the next chapter.

Chapter 3. What Are *p*-Cycles?

p-Cycles are a new protection mechanism for survivable networks. They offer many desirable properties such as capacity efficiency, fast restoration times, and flexibility in the routing of working paths [39]. As a preconnected ring-like structure, *p*-cycles have a very simple protection mechanism similar to conventional rings. However, by allowing straddling spans (spans that have their end nodes on the cycle but are not a part of it themselves) to be protected in addition to on-cycle spans, low redundancy levels can be achieved. This combination of features ("ring-like speed with mesh-like capacity" [40]) is the central value proposition of *p*-cycles and has been extensively discussed in many publications. *p*-Cycles have other interesting properties though, which we will also explore in the rest of this chapter.

Judging by the growing collection of literature on *p*-cycles, interest in this scheme appears to be reaching a critical mass with several different international research groups now having published articles in the area. This chapter, in addition to explaining the *p*-cycle concept, is a complete review of all known work on *p*-cycles from their discovery to the present time, and is thus one of the scholarly contributions of the thesis. We use the first section to explain the basic idea of *p*-cycles in an introductory way so the reader has a foundation for the historical overview that follows. We then begin chronologically (for the most part) to review the development of *p*-cycles along with some early publications. In examining more recent work, however, we will switch to a topic-oriented approach to consider similar publications and findings about *p*-cycles together. Here the papers are presented chronologically (in general) only within the discussion for each specific aspect.

3.1. Introduction to *p*-Cycles

As we saw earlier, mesh networks are very flexible and efficient, yet rings retain the advantage of speed because their cyclic structure permits the restoration action to be known prior to failure. This advantage is also shared by *p*-cycles, and as [41] explains, "the switching actions required in real time are completely preplanned and no more complex than that of a line-switched ring."

A *p*-cycle is simply a connected closed path of single (or multiple) channels in the spare capacity layer of a network. Because the *p*-cycles are formed out of unit-capacity single links, there can be multiple copies of the same cycle. This is often seen in actual solutions, where certain high-quality cycle routes are used repeatedly in forming many identical unit-capacity *p*-cycle copies. In a *p*-cycle based network, as in a mesh network, there is no structural association between the working and spare capacity usage – although working and spare channels may be physically built or "lit up" together as part of the same modular unit, the working routing of the demands does not place any direct constraints on the spare capacity structures, and vice versa. However, even though actual capacity may be provisioned in a modular way, there is no requirement for the *p*-cycle constructs themselves to have a modular structure, in contrast to conventional rings where a certain module size is a property of the ring [40].

Because *p*-cycles (in their basic form) are a span protection mechanism [42], there is no need to consider the end-to-end routing of working paths. When determining the set of *p*-cycles, only the working capacity values on each span need to be taken into account as part of the *p*-cycle design. This frees the working paths to be routed in any way desired.

The actual protection switching mechanism is very straightforward. Figure 3.1 shows a typical *p*-cycle along with the possible restoration scenarios. In Figure 3.1(a), no failure has occurred and the *p*-cycle is in a state of readiness. An on-cycle failure is depicted in Figure 3.1(b). In this scenario, the end-nodes of the failed span perform a simple loop-back switching action to direct the demand units normally traversing the failed span into the *p*-cycle instead. Because the *p*-cycle is preconnected at all other nodes, these are the only switching actions required. Note that in this case, the unit-capacity *p*-cycle can provide protection for one unit of demand on any of its on-cycle spans. Figure 3.1(c) shows the failure of a straddling span (one with both end-nodes on the cycle but not on the cycle itself). In this case, the same type of loop-back switching can be performed to effect restoration. This scenario is much more advantageous, however, because the unit-capacity *p*-cycle can provide two protection paths, one on each "side." This allows up to two units of working capacity to be protected on each straddling span. Finally, in Figure 3.1(d), we see the failure of a span "inside" the

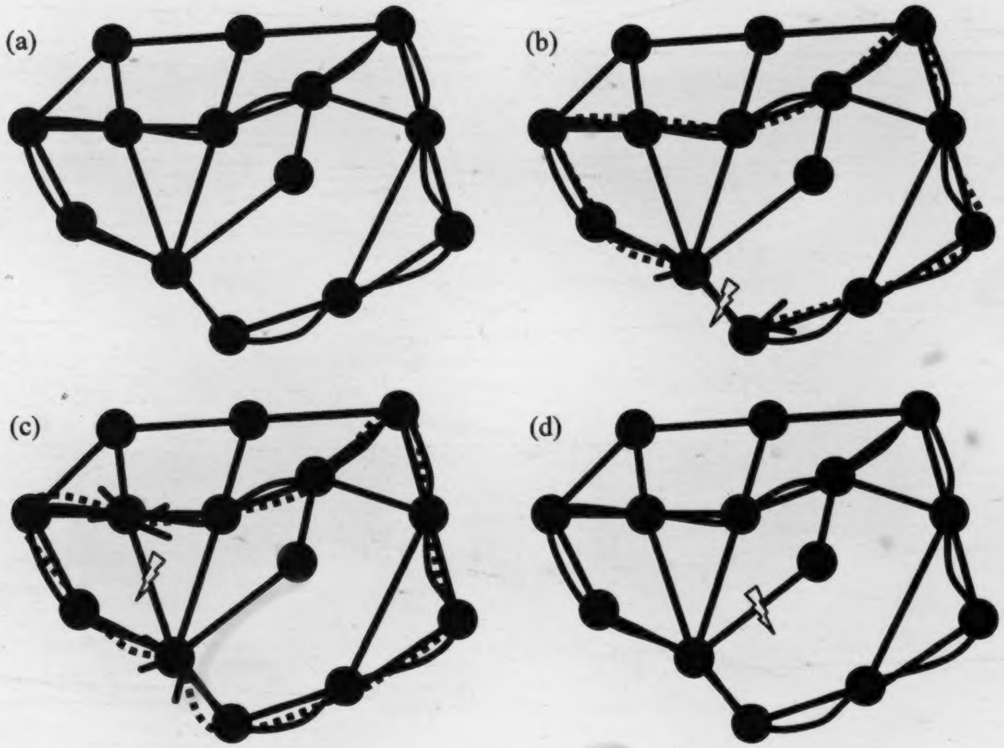

Figure 3.1. **(a) An example *p*-cycle, (b) the protection reaction for an on-cycle failure, (c) the reaction for a straddling span failure (with two restoration paths), and (d) an unprotected failure on an "inside" span which is neither on-cycle nor straddling.**

p-cycle, but not appearing in the conventional on-cycle or straddler configuration. For this failure, the *p*-cycle is unable to offer any protection.

It is relatively easy to see how a *p*-cycle can achieve fast, ring-like switching times, but what is not quite so intuitive is how *p*-cycles have the low redundancies that a number of publications have claimed. By inspecting our example *p*-cycle in Figure 3.1, in a logical hop-weighted framework, we see that the *p*-cycle has 11 hops, and therefore protects 11 on-cycle spans (for one unit of capacity each). It also protects four straddling spans for two units of capacity each. Therefore, the hop-count-based redundancy (assuming enough working capacity is present to take full advantage of the offered protection) is $11/\left[11(1)+4(2)\right]=11/19=57.9\%$. This is significantly less than a ring, which would have 100% redundancy in the same hop-count framework (assuming it was fully loaded). This may not always be possible in practice because of stranded capacity issues, and therefore actual (as utilized) ring redundancies can be much higher [29].

3.2. Spare Capacity Preconnection

We now travel back to 1994, before the discovery of *p*-cycles. As an initial short paper in *Electronics Letters* [43] explains (citing [44]), digital crossconnect systems (at that time) were said by some to take an unacceptably long time to conduct a closure (switching) operation, precluding the real-time connection of restoration paths. An idea that was explored in that work ([43]) was thus to examine if the spare capacity channels in the network could be preconnected in some optimal fashion so the network would be in the most "ready" possible state when a failure occurred. It was recognized, of course, that some connections would need to be made in real-time, but to the extent that some would already be found in a connected state at the time of failure, this might allow a faster overall recovery. The finding of this preliminary study was that "an average of 79% of the span-to-span pairings required for restoration in the 10 test networks could be preconnected in advance of failure" [43]. The concluding remarks went on to state that there was thus a motivation for conducting further research on preconnection strategies.

A Canadian patent application ([45]) was filed in the following year, which also dealt with the preconfiguration issue. The idea was essentially "a method for restoring traffic in a network," [45] with three steps: (1) finding the restoration routes for each span failure, (2) finding the amount of flow along each route to minimize the total unrestored traffic (for all span failures), and (3) preconnecting the network in accordance with a result from the previous step [45]. A similar U.S. patent was also filed and has since been issued [46].

These three publications were followed up by a paper in the *Journal of Network and Systems Management* [47]. This went beyond the initial assessment of potential in [43] to consider the actual link-to-link connections required in the preconfigured solution. As [47] explains, the previous study was "only an assessment of the upper bound achievable by preconfiguration as the calculation was based on the number and pattern of cross-connections required at each node in isolation." This paper considered the actual channel-level connections required to form paths, and found that network readiness values (the percentages of working links that could be restored with a preconfigured path) were in the order of 30-40%.

3.3. Cycle-based Preconfiguration

3.3.1. Discovery of Preconfigured Cycles

This work on preconfiguration in general was a foundation for investigations on the topic presented in the M.Sc. thesis of Demetrios Stamatelakis [48]. Initially, preconfigured spanning trees were considered, however, it was concluded, "The tree pattern type appears to have a relatively poor performance when applied to preconfigured restoration" [48] (p. 46). The next approach was to use a genetic algorithm (GA) to find patterns that were not restricted by the nature of the pattern type. For example, cycles and trees have certain properties, which must be present for a construct to be called a cycle or a tree. The idea in the GA approach was to move away from this, to be able to consider any pattern, as long as it scored well in the GA's selection criteria. The chapter on the GA approach concluded that the restoration of the GA patterns "was quite effective compared to the performance of the preconfigured tree algorithms" [48] (p. 66). But interestingly, as a historical account in [14] (p. 664) explains, "Results were saying that even *without* adding any spare capacity above the strict minimum for span restoration we had reached 100% preconfiguration and...*all the patterns were cycles!*" The next chapter in [48] considered the design of cycle-based protection using Integer Programming techniques, likely because cycles were in many cases (but actually not always) appearing as a promising output pattern from the GA, and also because a cycle is a pattern that could be realistically implemented with a DCS machine ([48] (p. 71) explains that in a DCS, there is no provision "to directly connect together more that [*sic*] two port cards"). It was found that for full preconfigured restorability in the test networks considered, a –3.81% to 18.54% premium in spare capacity would be required over the original sparing plan (although for the result with the higher 18.54% premium, this was not a strictly optimal solution, but only the best that could be obtained before the memory limits of the computer were reached [48] (p. 83)).

3.3.2. Method for Distributed Preconfiguration

A final development in [48] was a distributed preconfiguration algorithm for finding cycles. This is based on the more general concepts of self-healing and self-organizing networks as outlined in [23],[24]. Drawing on both sources, a high-level view of self-organization can be presented by first describing a "statelet." The best way to think of a statelet is a shared memory field on a network link. It is carried in the overhead of the transmission system, and is not a one-time "message," but simply a small set of data fields that are continuously asserted over the link (until changed). By changing outgoing statelets and observing changes to statelets on incoming links, network nodes can communicate with each other to effect the discovery and connection of preconfigured cycles (or more generally, as in [23], failure restoration paths). With an appropriate algorithm running in each node, no central control or supervision point is required, and the nodes can autonomously interact to implement the set of protection cycles. In what [48] (p. 100) describes as the DPCD algorithm, nodes can have one of two roles: cycler or tandem. The cycler node initiates a flood of statelets that are propagated through the network by the tandem nodes. These nodes apply a set of rules to allow statelets with high "scores" to take precedence over other statelets when competing for access to the outgoing links. In this way, the tandem nodes influence the pattern of statelet propagation, and ultimately, the cycles that are formed. Eventually, each statelet will be seen again at the cycler, which will recognize the statelet as one of its own and therefore carrying the description of a cycle route as encoded on a node-by-node basis by the tandem nodes comprising the cycle. As different statelet "trails" find their way back to the cycler node, a record can be kept of the best cycle seen so far. When no more cycles can be found, the best one is constructed and the process is repeated with the next node in sequence as the cycler.

3.3.3. Associated Patents

In a Canadian patent application [49], a number of these developments are discussed. The patent claims not only the general idea of forming closed, preconnected protection cycles, but also the distributed cycle-finding method with statelets described above. A similar U.S. patent has since been issued [50].

3.4. Subsequent Initial Developments

3.4.1. First *p*-Cycle Publications

Although significant progress had been made in the development of these *p*reconfigured *p*rotection cycles, the idea had not yet been disseminated to a wider audience. The first publication on the topic was [51], which discussed OPNET simulation of the distributed mechanism described earlier, now renamed "DCPC" and extended to include a global comparison phase between all cycler nodes before the construction of a cycle, as opposed to constructing one after exploration by each cycler node. This paper was also significant in that it contained the first use of the term "*p*-cycle."

The seminal paper on *p*-cycles though, was [40], at ICC '98. This covered the *p*-cycle concept and its benefits, the optimal design of *p*-cycle networks including the IP formulation and results, along with a discussion of self-organizing methods, the DCPC protocol, and results from the OPNET simulation as well. This is one of the papers very frequently cited in a general background discussion on *p*-cycles.

A subsequent paper at the 1998 Canadian Conference on Broadband Research [39] also covered the same topics, with more detail in some of the introductory portions.

3.4.2. Node-Protecting and IP Layer *p*-Cycles

The next area of *p*-cycle development was in protecting against router (i.e. node) failures in IP (Internet Protocol) networks. Here, virtual *p*-cycles are configured (using a virtual circuit structure, such an MPLS label-switched path) to logically surround each node in the network. In the event of node failure, packets that would otherwise traverse the failed node can use the node-encircling *p*-cycle as a detour path. Of course, IP layer *p*-cycles (which are, in general, not required to be node-encircling) can be used for protection against span failures as well. An advantage of *p*-cycles in the IP layer is that they require no spare capacity in the prefailure state, because they only exist as logical routing table (or label-switching) entries. They are also advantageous because they can offer quick restoration before conventional IP layer methods have time to react. These factors were all discussed in [52], which was the first paper on the topic.

Prior to [52], however, Canadian and U.S. patent applications [53],[54] were filed. A more comprehensive journal article on IP layer *p*-cycles was later published as well [55].

3.5. Refinements and Dissemination

After the articles on IP applications, a few articles that were more general in nature appeared to discuss and disseminate some of the concepts and methods that were known thus far. Another one of the "famous" papers on *p*-cycles is [29], an invited review paper at the DRCN 2000 conference. It touched upon most known aspects of *p*-cycle work, including the concept itself, optimal design in a SONET or WDM framework, self-organizing methods, and IP layer techniques. Moreover, it also publicly introduced and explained the "capacity-slice" ADM-like *p*-cycle node structure.

The next paper to appear [56] was a short treatment of some theoretical considerations. The authors were able to show that *p*-cycles "have as high a restoration efficiency as it is possible to expect for any type of preconfigured pattern" [56].

Following this, a TRLabs technical report [57] touched on varying aspects ranging from early work with preconfigured spanning trees and genetic algorithms to Integer Programming formulations, in addition to the theoretical considerations just published.

Finally, a review article in *IEEE Communications Magazine* [19] included *p*-cycles in the collection of topics presented, along with the vision of a network where "guaranteed and best-effort protection services are provided to each working path by an otherwise unseen but self-organizing adaptive layer of *p*-cycles" [19].

These marked the end of the first "phase" in *p*-cycle development by researchers in the Network Systems group at TRLabs. A majority of the papers immediately following this period came from researchers in other groups around the world, as we will soon see. During this period many efforts were also made through software and presentations to transfer the technology to Nortel Networks.

3.6. WDM Considerations

An interesting area in subsequent research is the use of *p*-cycles in a WDM context where factors such as wavelength conversion and cycle length limits come into play.

This area of investigation was spearheaded by the Institute of Communication Networks at the Technical University of Munich (TUM).

3.6.1. Wavelength Conversion Effects and Initial Studies

The first work to consider the application of *p*-cycles in the WDM environment was Claus Gruber's Diploma thesis at TUM [58]. The differences in *p*-cycle design in the Virtual Wavelength Path (VWP) and Wavelength Path (WP) cases (corresponding to wavelength conversion and no wavelength conversion, respectively) are explained, and optimization models for both scenarios are presented. It was found that while VWP networks require less spare capacity, in the WP case "the ratio between spare and demand wavelengths is still quite good" [58] (p. 74). Gruber's thesis also proposed an adapted metric for working path routing (which was found to increase the number of working wavelengths, but reduce the number of wavelengths used overall), and the idea of non-simple *p*-cycles (the use of which was found to reduce the redundancy and increase flexibility when attempting to find a solution with a cycle length limit in effect).

This was followed by [59] at ICC 2002, which includes many of the same topics. VWP and WP redundancy results for various cycle length limits were presented, and in particular, this paper was the first by a group outside TRLabs to independently validate the claims of low *p*-cycle redundancies that appeared in prior articles. Notably a fully restorable European network (COST 239) was reported using a *p*-cycle design requiring only 34% redundancy.

A review article that came shortly thereafter [60] covered many of the basic *p*-cycle concepts, including the process of obtaining an optimal design solution, along with case study results (from [59]). It also discussed (in a general way) how incremental and global re-optimization of the *p*-cycle set would be done. A subsequent invited review paper [61] also treated the *p*-cycle concept and briefly outlined some of the results from [59] as well.

In [62] (from a researcher in yet another group), a slightly different approach is taken in that only one working fiber per span is assumed instead of the multiple fiber approach used in previous work. With this assumption in the WP case, the exact routing of working demands becomes very important. This paper also assumes a unidirectional orientation for the *p*-cycles (which was also mentioned as a possibility in [58]). A simple heuristic is presented which performs both wavelength routing and *p*-cycle selection. The paper describes this heuristic as being "very similar to the 'First-Fit' assignment of wavelengths" [62], and presents case study results comparing the VWP and WP cases (with the heuristic) over a range of maximum allowed *p*-cycle sizes.

3.6.2. Partial Wavelength Conversion

The next area to be explored in the WDM *p*-cycle area was the effect of partial wavelength conversion, instead of the all-or-nothing capability previously assumed. This was initially presented in the Diploma thesis of Matthias Scheffel at TUM [63]. (The possible usefulness of partial conversion was also mentioned in passing in the conclusions of [62] as well.) However, before presenting the partial wavelength conversion studies, [63] included a comparison of *p*-cycles to dedicated (1+1 APS) and shared (SBPP) path protection mechanisms. It was found that while 1+1 APS was the most costly and SBPP was the cheapest, *p*-cycles were only slightly higher than SBPP.

Two different cases were considered in [63] for partial conversion. In both scenarios, it was assumed that the working capacity layer was WP (i.e., without conversion), requiring each demand to use a single wavelength (or "color"). In the first case, the *p*-cycle layer was also WP, but a pool of wavelength converters (WCs) was made available as a demand transited from the working layer into a *p*-cycle (and vice versa). In the second case, the *p*-cycle layer was fully VWP allowing the *p*-cycles to be composed of multiple colors. Results were obtained for the COST 239 study network, and the conclusions stated that partial wavelength conversion was very close in terms of efficiency to the case with full wavelength conversion [63] (p. 80). It was shown that the number of wavelength converters required is minimal, and that "Typical values are less than 10% of the total amount of WCs in the pure VWP case" [63] (p. 80). An analysis of the WC cost tradeoff showed that as WC cost increases, fewer of them are used with more *p*-cycles being deployed to compensate. Another interesting result was that a significant capacity reduction was obtained by introducing a small number of converters, with further converters having an increasingly smaller benefit. The comparison of the two cases initially proposed for study showed that the strategy of placing WCs on the *p*-

cycles instead of only at the access points "needs more than twice as many WCs while providing about the same protection-efficiency" [63] (p. 74).

Some of the findings in [63] were then subsequently published in a conference paper at DRCN 2003 [64], which highlighted the fact that placing the converters only at the *p*-cycle access points (as opposed to having fully VWP *p*-cycles) was an efficient and simple strategy. A revised and extended journal version was published in [65], with more detail and presentation of the complete optimization models.

3.7. Hamiltonian Considerations

It was first suggested in [56] "that if Hamiltonian cycles exist, these may make good p-cycles." A Hamiltonian is a cycle that passes through every node in a network exactly once, and in the *p*-cycle framework, this means that every span will either be on-cycle or straddling the Hamiltonian. The idea of using Hamiltonians as *p*-cycles has been explored and extended in a number of subsequent papers, which we discuss next.

3.7.1. Hamiltonian Cycle Protection (HCP) and Related Methods

A scheme called HCP was proposed in [66], and although the *p*-cycle terminology is not used, the central idea in HCP is to protect the network with what appears to be a single Hamiltonian *p*-cycle. By using only one cycle for the entire network, straddling spans must obviously be protected by that cycle as well, leaving little doubt that the cycle must be functioning in the same way as a *p*-cycle (as opposed to a conventional ring, which cannot protect straddling spans). However, in their efficiency calculations, the authors seem to miss the fact that each straddling span can have two units of protection instead of just one – the straddling spans are being protected, but not to the full extent that they could be. They do present a brief discussion of *p*-cycles in their appendix and recognize that the *p*-cycle scheme "can potentially derive our result in the limiting case" [66]. It is important to realize though, that in spite of the different nomenclature, the HCP method is a trivial special case of the conventional *p*-cycle scheme – essentially, just using one big *p*-cycle.

A follow-up paper from the same authors [67] suggests two new schemes to extend HCP. It also discusses how to provision capacity on a cycle in HCP when the network is non-homogeneous (variable working capacities on spans), and this discussion now recognizes that straddlers can indeed have two working channels. The first extension proposes dividing the network into domains and using a single Hamiltonian cycle to protect each one. Aside from the terminology confusion (a cycle isn't a Hamiltonian if it doesn't cover the whole network), this initially appears to be just a transition from one type of heuristic method (HCP) to another (called Hamiltonian Cycle Cover, or HCC). As initially proposed in HCP, one *p*-cycle would be placed for the whole network, and here (in HCC) a set of *p*-cycles (at first glance) would be tiled across its regions. But the paper goes further to fundamentally alter the nature of the cycles in HCC by provisioning only the maximum spare capacity required by either of two cycles passing over a common span. This sharing of spare capacity makes HCC no longer a *p*-cycle method, nor really a cycle method at all, because the spare capacity on common spans is shared between them. The second modification, Hamiltonian Cycle Neighbor Capacity Closure (HNC), is a further attempt to reduce the capacity on common spans by using "cycles" in neighboring regions together to restore a single span failure, resulting in a reduction in capacity needed on the shared span. However, this moves even further away from a stated motivation of the extensions, namely, to provide dual-failure robustness (which will decline when cycles in multiple regions are used for restoration of a single failure).

These methods were subsequently presented in a review article [68].

3.7.2. Restricted *p*-Cycle (RPC) Method

A similar method (i.e., using a single Hamiltonian *p*-cycle) was also presented in [17], and in this case the authors do use the *p*-cycle terminology. A simple optimization model for finding a Hamiltonian cycle is presented, along with a complexity comparison between *p*-cycles and RPC for the extreme upper limit of a fully-meshed network. Included in the performance results is a graph showing the restorability level for both methods as the available spare capacity is increased. Another (later) article on RPC considering Quality of Protection (QoP) factors [69] compares the recovery times and end-to-end delays in both RPC and dynamic restoration.

3.7.3. Hamiltonians in Semi-Homogeneous Networks

Motivated in part by the feeling that the works above had a somewhat weak and confusing influence on the field, a paper in *IEEE Network* [70] (containing work done by the author of this thesis and his supervisor, and found later in Chapter 4) reviews and develops some further issues regarding Hamiltonian *p*-cycles, including a comparison of capacity requirements in single-Hamiltonian and conventional design, and a demonstration that, if used, a Hamiltonian *p*-cycle can reach the theoretical lower limit on redundancy in a span-restorable mesh network. A proof is also given that a Hamiltonian cycle may not always produce the highest value of the *a priori* efficiency metric (to be defined in the next section).

3.8. Cycle Preselection and Cycle-Finding Heuristics

The area of work dealing with algorithms to manipulate cycles outside of the ILP solver environment (other than possibly some simple heuristics for finding or choosing Hamiltonians) can be separated into two streams. First we look at the idea of preselecting *p*-cycle candidates to be used as input to an ILP solver, and then at a heuristic method which proceeds iteratively to develop a complete multiple cycle design.

3.8.1. Preselecting Candidate Cycles for an ILP Solution

In the design of optimal *p*-cycle networks, like conventional mesh networks, a preprocessing step is employed to format and/or generate the input data (nodes, spans, candidate cycles, etc.) for the ILP optimization engine. Similar to the phenomenon in mesh design where the space of unrestricted eligible working and restoration routes explodes dramatically as the network size increases, the set of eligible cycles is also exponential with increasing network size. This creates a problem when performing an ILP-based *p*-cycle design. On one hand, we would like a restricted set of candidate cycles so the ILP solver is not overloaded with possibilities (lengthening the solution process and possibly preventing it from finding any solution at all within a reasonable time frame). On the other hand, by restricting the cycle set, the solver is not given the full universe of choices to consider, and may therefore return with a sub-optimal solution.

A simple strategy to limit the input cycle set is to restrict it to being composed of a certain number of the shortest cycles in the graph, for example, the 500 shortest cycles. This approach has been used in some of the *p*-cycle work discussed so far. However, it is somewhat shortsighted when we consider some of the interest in Hamiltonians we saw earlier (and more generally, large cycles that may not be strictly Hamiltonian). It may be that the efficiency of a *p*-cycle is not always that high for a shorter cycle, and by selecting these shorter cycles exclusively, we are unduly impairing the quality of our solution.

Motivated by this, in [71] two preselection heuristics are explored, which provide a way to strategically limit the input cycle sets. These heuristics are based on selecting cycles on the basis of a metric instead of simply their length. One of these metrics, topological score (*TS*), considers the total number of working capacity units that a certain cycle could protect. The other, *a priori* efficiency (*AE*), does the same, but is normalized to the cycle length (or more generally, any other measure of cost). It was found (in [71]) that a solution very close to optimal could be obtained by using a small candidate cycle set selected with the *AE* metric, and further, that the ILP solution time with the limited set was significantly faster as well. Of note also is that [71] was the first paper to report results from a joint formulation (where the working routes and *p*-cycles are optimized together), and that it reported large benefits from the joint optimization.

A greatly different cycle-finding approach was proposed in [72], called the Straddling Link Algorithm (SLA). Instead of using an exhaustive cycle search followed by selection of a subset (as in [71]), SLA is a heuristic that considers each span and forms a cycle around it, preferably as a straddler, but as an on-cycle span if necessary. Therefore, as the conclusion to the paper points out, SLA will provide a set of candidate cycles no larger than the number of spans in the network. However, [72] did not include any ILP results using the SLA candidate cycle sets, leaving us with no indication of their overall quality. Such single-straddler *p*-cycles are also prima-facie of very low potential efficiency, so it is hard to understand what the particular advantage of this approach was to ever be.

This area of *p*-cycle design, the selection and strategic limitation of candidate cycle sets, is an interesting direction for future work – we cannot escape the fundamental laws governing the size of unrestricted cycle sets, nor can we dramatically improve the performance of ILP solvers (beyond the usual strategy of buying a faster computer), leaving the cycle preselection strategy as one variable that we can still control. We return to look at this further in Chapter 7.

3.8.2. Purely Algorithmic Methods for *p*-Cycle Network Design

As [18] explains, the Straddling Link Algorithm does not (in general) produce cycles that are very efficient, because they can only have at most one straddling span. The focus in [18] is thus to use the SLA *p*-cycles as a starting point to evolve a better and more efficient cycle set, but still with a heuristic method. After reviewing the simple "Add" and "Join" operations proposed in [14] (pp. 708-712), the paper goes on to propose more advanced "SP-Add," "Expand," and "Grow" operations for increasing the size and efficiency of the set of heuristically-determined *p*-cycle candidates. Two variants (corresponding to either the "Expand" or "Grow" operations) of an algorithm called CIDA (Capacitated Iterative Design Algorithm) for finding a fully restorable *p*-cycle design are then presented. Cycles with the highest "working-weighted" (actual as opposed to *a priori*) efficiency are placed iteratively until all working capacity is protected. This creates a set of *p*-cycles that are most effective (in an iterative, "greedy" way) at protecting the working capacity present as a result of a specific demand routing. It was shown that while the solutions produced were still higher in redundancy compared to optimal (although the variant using "Grow" was close), the run times were a fraction of those for solutions from the optimal ILP solver with a large set of candidate cycles.

3.9. Ring Mining

The concept we call "ring mining" is not strictly a *p*-cycle idea. The basic idea and motivation involves using existing capacity (and possibly nodal equipment as well) from a legacy ring network as the building blocks for a network with a different, and likely more efficient, protection or restoration method. *p*-Cycles are, of course, one such method.

The first place where this idea was suggested for *p*-cycles (in a subtle but inherent way) was in a Canadian patent application [73] (and the corresponding U.S. patent [74], now issued). They describe a device that can be attached to an existing ring ADM to allow the node to then function as a *p*-cycle node with access to straddling span(s). This would allow existing ring equipment to be re-used in conjunction with these new devices to form *p*-cycles. A section in [29] also gave a brief description of the add-on unit.

Ring mining was more fully developed and described later, first for the generic span-restorable mesh case in [19],[75]. Canadian and U.S. patent applications to follow ([76],[77]) covered the generic case as well and presented three approaches to the idea, but also included *p*-cycles as an option for migration as well. Ring mining was then briefly discussed (along with other topics) in a short review paper [78].

However, the main work on ring mining specifically related to *p*-cycles was [79]. It provides a good background discussion before explaining the add-on device for *p*-cycle ring mining (now called a Straddling Span Interface Unit, or SSIU), as previously proposed. An example is then presented of ring to *p*-cycle evolution, which illustrates how the efficiency of the network can improve as the migration progresses. Optimization models along with a case study for a Canadian metro network are given next and the results showed demand growth multipliers of up to 2.75 times for some networks, with no new capacity additions. Other results showed the required new capacity to achieve a range of target growth multipliers with and without adjusting the existing working routing. For the TELUS case study network, it was found that a 1.5 times demand multiplier could be achieved with no extra capacity. But "for a relatively minor investment" [79] above the capacity currently present, the demand carried by the network could actually be doubled. The conclusions in [79] stated that as a target architecture, *p*-cycles are nearly as efficient as a span-restorable mesh.

3.10. Extensions for Path Segment Protection

In the entire collection of *p*-cycle work thus reviewed, it can be noted that the *p*-cycle mechanism is analogous to span restoration in a mesh network, in the sense that when a span failure occurs, the failed demand units are rerouted (using *p*-cycles) only between the end nodes of the failed span. An interesting question is to speculate on whether path-protecting *p*-cycles might be viable, and if so, how they might operate. This is the issue addressed by the papers in this section.

The first article ([42]) introduces what it calls "flow *p*-cycles," which are *p*-cycles that protect transiting demand units over an arbitrary path, instead of only over on-cycle and straddling spans. Note that this arbitrary path does not have to be end-to-end, but can be any path segment, with the whole path of course being a special case. Such segments can be on-cycle or straddling to each *p*-cycle in the same way that individual spans are. After explaining this, the paper presents a design model and results showing that flow *p*-cycles can achieve a capacity savings relative to span (i.e. conventional) *p*-cycles, and that flow *p*-cycles are also quite close to mesh path restoration without stub release [42]. The idea of using both types of *p*-cycles in a network is presented as the paper concludes.

An extended journal version [80] also presented the concept of flow *p*-cycles and the ILP model, but additionally proposed a new preselection scoring metric, a design model for complete span *and* node failure protection, and a model for maximizing the node recovery level within a certain amount of already provisioned spare capacity. It was found that node failure protection could be provided with only a small capacity premium over traditional span-protecting *p*-cycles. Some implementation issues and the idea of using span and flow *p*-cycles together to each protect certain demands were covered near the end of the paper, in addition to a very brief discussion of using the "protected working capacity envelope" idea (covered in the next section) with flow *p*-cycles.

3.11. Further *p*-Cycle Ideas

In this section we briefly touch on a few ideas for *p*-cycle design that may not neatly fit into one of the other categories, but are still interesting or useful on their own. Most of these appear in only one or two papers, and some are still areas of ongoing work.

3.11.1. Directed and Non-Simple *p*-Cycles

In [58], the idea of unidirectional *p*-cycles was proposed to cope with uneven demand distributions. Unidirectional *p*-cycles were also discussed later as part of a book chapter [81]. After reviewing *p*-cycles and the basic design model, the chapter goes on to present the (rather confusingly named) "routing of flows and slacks." Although it is somewhat unclear, it appears the method is a joint formulation involving unidirectional *p*-cycles. Some further analysis and comparative results are also given.

An interesting paper on non-simple *p*-cycles [82] appeared shortly after this. The author outlines several types of non-simple *p*-cycles and presents a few examples of their versatility over conventional simple cycles used in the majority of *p*-cycle designs. A joint ILP optimization model for working routing and non-simple *p*-cycle placement is given followed by a case study which showed that in some cases there can be a small capacity savings achieved by using certain types of these non-simple cycles (the savings were more pronounced when stricter circumference limits were imposed).

3.11.2. Forcers and the Protected Working Capacity Envelope

Of note is the allusion in one of the figures in [60] to what will later become known as the "protected working capacity envelope" (PWCE) concept (discussed in more detail for the conventional span restoration context in [14] (pp. 282-291),[16]). In general, in the PWCE framework, incremental working demands are routed in such a way that they can be protected by the current configuration of the protection or restoration method in use. In a *p*-cycle network (as [60] suggests), we could check if an incrementally added demand could be protected by an existing set of *p*-cycles, and route it accordingly. (And of course, in cases where protection is not possible, the *p*-cycle set would be adjusted.)

Closely related to the PWCE concept is the idea of forcers [32],[33] (previously discussed in Section 2.2.4). In [83], the forcer concept is applied to *p*-cycles. A new model is proposed to allow the quick calculation of all forcers and their magnitudes (now in the context of *p*-cycle design) in one step, instead of the more tedious methods presented previously in [32],[33]. The results from this forcer calculation model "show that typically quite significant numbers of extra working channels can be used with no increase in spare capacity budget" [83]. Two additional models are given to maximize the protected working capacity in a *p*-cycle network with either a total spare capacity budget or a set of spare capacity values on each span. These also definitely had an effect and offered redundancy figures that were lower than the conventional reference designs. In addition to these models, the paper offered an explanation of hop-based and distance-based lower bounds on redundancy, and why the distance-based figure can sometimes go below the lower bound in a logical hop-based framework.

3.11.3. Dual-Failure Survivability

As [84] points out, "Dual-failure situations, or the equivalent, are not as rare as might be thought in an extensive national or regional optical network." The first look at dual-failure restorability for *p*-cycles appeared in [85], which begins by reviewing all the possible dual-failure cases in a *p*-cycle network. A bi-criteria objective approach is proposed, which allows either the total number of *p*-cycles or the total quantity of protected working capacity to be maximized in conjunction with spare capacity cost. By changing the weight of the bi-criteria term, different emphasis can be given to both factors. Results from a case study reviewed next showed that longer *p*-cycles (as might be expected) had a detrimental effect on dual-failure restorability. Depending on whether the number of cycles in the design was minimized or maximized, dual-failure restorability levels were reported as 50% or (approximately) 70%, respectively [85]. What appears to be a subset of the results in [85] was then presented in [86]. This subsequent paper [86] also offers a calculation for the spare capacity lower bound for dual-failure restoration with a span-restorable mechanism.

A more theoretical treatment of availability and dual-failures in *p*-cycle networks appeared in [87] (pp. 143-156). This chapter in a Ph.D. thesis about availability in mesh networks started (after a review of *p*-cycles, of course) by covering the dual-failure outage causing scenarios, path unavailability, a comparison of path unavailability for on-cycle spans and straddlers, and a study which compares *p*-cycle performance to conventional span restoration for dual-failure scenarios. As the chapter succinctly points out, "Results show much higher restorability levels for the span restoration mechanism" (than for *p*-cycles) [87] (p. 153). It then goes on to explain that the difference is mostly because of the large *p*-cycle sizes in the test designs, which are more likely to be selected as part of a capacity efficient solution but are not capable, on an individual basis, of handling dual failures, which obviously attack these larger cycles more often because of the higher number of spans that rely on such *p*-cycles for their protection. Other results showed a much wider distribution of dual-failure restorability levels for *p*-cycles compared to span-restorable mesh, where values for the majority of paths were clustered at the upper end of the scale.

3.11.4. Mixed Design Approach with Other Methods

An interesting question is the effect that *p*-cycles might have when introduced into a suite of protection and restoration methods that provide different grades of protection to different types of demand in a network. In [21] (which presents this author's work documented later in Chapter 5), it was found that although there was not a significant capacity savings, a surprising and important result was that many demand units can "move up" to enjoy a faster method of protection than they would otherwise be assigned. This suggests that in networks where survivability is implemented with a slower mechanism, like SBPP, there is perhaps a hidden benefit waiting to be exploited – that is, allowing certain priority demands to use a faster method but at no added cost.

3.11.5. MPLS Layer *p*-Cycles

Further investigation into MPLS layer *p*-cycles beyond that done in the early work at TRLabs was presented in [88]. In this very interesting paper, the authors define a model for *p*-cycle design in the MPLS layer where the concept of a unit channel size does not exist. They also present a thorough side discussion of cycle search strategies and their finding that when working load is balanced, large *p*-cycles are effective, but when load is unbalanced, small *p*-cycles are required as well for an efficient solution. The authors are thus able to propose a selection heuristic that chooses only the largest and smallest cycles for the input candidate set. The paper ends with a realistic case study on a European network. The concluding discussion points out the positive effects of load balancing and the proposed MPLS scheme, stating that its use in the example scenario "has demonstrated that our scheme can achieve very good performance without intensive computation" [88].

3.12. Overview Papers and Comparative Studies

3.12.1. General Comparison Articles

In [89], a simple comparison was performed for different survivability schemes under either or both of two switching architectures – electrical-optical and all optical. The methods considered were dedicated protection (which, as presented, seems to be 1+1 dedicated APS), span-restorable mesh, and *p*-cycles. As the conclusion states, "The P-cycle approach had fewer OEO conversions and used fewer wavelength kilometers than the dedicated model without suffering a significant availability penalty. P-cycles were found to be nearly as effective as mesh algorithms at reducing bandwidth requirements while requiring 46% fewer OEO conversions" [89]. This report is yet another verification from an independent source of the attractiveness and efficiency of *p*-cycles.

Another comparison between protection and restoration methods was presented in [90]. After a good overview discussion, two comparisons (with a common test network and demand pattern) involving *p*-cycles were given, the first being the spare capacity required to recover from a single span failure. *p*-Cycles were the most efficient of the six methods considered. For the other aspect however, *p*-cycles were the worst method, having the highest mean percentage of lost connections in a dual-failure situation. (In general we can attribute this to the high amount of spare capacity sharing typically seen with *p*-cycles, and the non-adaptive protection mechanism. For any second failure event in a dual-failure pair, in comparison to more adaptive mechanisms like span restoration for instance, the odds are greater that a necessary *p*-cycle is already being used to restore traffic for the first failure and is therefore unavailable.) In general, the relatively poor dual-failure performance reported in this paper is also supported by the dual-failure results for *p*-cycles given in [87] (pp. 153-155).

A follow-up paper [91] also considered a suite of methods and this time compared them in the framework of dynamic demand. Profiles of traffic scaling factor versus blocking probability and network load were given for each method considered on two test networks. The results showed that *p*-cycles were generally (except at large scaling factors) either comparable or slightly lower than other methods for network load, and had, for one network, either (comparatively) excellent or poor blocking probability depending on the scaling factor (high or low, respectively). For the other network, at a low scaling factor, blocking was either superior or quite poor, in this case depending on the method for adapting to the dynamic traffic (whether the routing of the demand is done in unallocated overhead or in overhead plus unneeded spare capacity scavenged from a *p*-cycle rearrangement). At a high scaling factor in the second network, both dynamic *p*-

cycle methods did well. The study also concluded that the ILP approach that was used would be suitable for dynamic *p*-cycle design because it normally solved quite quickly.

In [92], a number of ILP models were presented for a collection of various survivability methods, including *p*-cycles. Comparative results were also given for a pair of test networks.

3.12.2. Confusion of *p*-Cycles with Similar Methods

Other cycle-based protection methods have been recently proposed in addition to *p*-cycles, these being "enhanced rings" in [93], "orientable cycle double covers" (OCDCs) in [94], and "generalized loop-back recovery" in [95]. An invited paper [41] takes as its mandate the clarification of the differences between enhanced rings, OCDCs, and *p*-cycles. In particular, the paper notes how *p*-cycles are the only method to offer straddling span protection. The generalized loop-back method does not incorporate the straddling span concept either, and as [14] (p. 135) states in regard to OCDC and generalized loop-back solutions, "In each case the protection is by a direct covering of cycles." The idea of straddling spans that do not need to be covered themselves with any protection capacity is truly a hallmark of *p*-cycles and can be used as a quick "test" to see if other methods are equivalent – those covered in this section, however, are definitely not.

3.12.3. Additional Overview Presentations

The *p*-cycle idea was included as one of the networking options presented in a short review paper at OFC 2003 [78]. However, this was only a very brief overview of the concept itself. A much more comprehensive discussion appeared in the chapter on *p*-cycles (pp. 659-748) in the *Mesh-Based Survivable Networks* textbook [14]. Topics covered included the *p*-cycle concept, basic properties, historical development, optimal capacity design, WDM considerations, heuristic and algorithmic approaches, domain perimeter (flow) *p*-cycles, Hamiltonian *p*-cycles, the SSIU device, self-organization, and MPLS layer and node-encircling *p*-cycles. While obviously not able to include full details for all known *p*-cycle techniques, this chapter in [14] is still an excellent resource.

3.13. Summary

Upon concluding our comprehensive review of the entire body of *p*-cycle literature, we can reflect on those that made an exceptionally strong contribution and might serve as an initial reading list for someone new to the field.

The first two that stand out vividly are [29] and [40]. These give a comprehensive foundation in the elementary *p*-cycle techniques, with an excellent summary of the initial "phase one" advancements in the area. Another valuable paper is [55], for its presentation of the Internet Protocol applications and node-encircling *p*-cycles. The basic foundation paper on WDM effects (and probably one of the most referenced) would have to be [59]. A significant issue in *p*-cycle design, which impacts scalability, is candidate cycle preselection, making [71] and [88] important to read as well. For a background in algorithmic techniques, the best overview and treatment can be found in [18]. The chapter on *p*-cycles in [14] (pp. 659-748) is also a comprehensive and worthwhile source. And finally, to return to the basic definition of *p*-cycles and to reinforce some of their advantages and unique properties, the comparison in [41] with other cycle-based protection schemes is a good way to wrap up a list of especially helpful articles.

In recent years the publication count on *p*-cycle topics has increased significantly and so it is somewhat natural to expect that, in spite of this tremendously rich collection of *p*-cycle knowledge we have already, that it will only continue to grow and diversify in the years ahead (as evidence, even as this chapter was being prepared, four more very recent publications [96],[97],[98],[99] became known). This chapter represents a scholarly contribution of this thesis and serves as a "snapshot" of where the field stands today, at a point where *p*-cycles are attracting increasing interest from both industry and the academic community. We now move on to the research-related chapters of the thesis.

Chapter 4. Hamiltonian *p*-Cycles: Addressing Some Confusions, Providing New Insights

Hamiltonian cycles were first identified in [56] as good possible candidates to be used as *p*-cycles. More recent interest from groups at the Georgia Institute of Technology (in [66],[67],[68]) and the Information and Communications University in Korea (with two papers, [17] and [69]) suggested it was timely to explore the idea in more depth. Most of the discussion and results we cover below were first published by the author and his supervisor in [70] and are also based on excerpts and extensions from the corresponding section in [14] (pp. 719-726).

4.1. General Background

Hamiltonian cycles, of course, have the special property of visiting all nodes in a graph exactly once [14] (p. 177). There is no guarantee that a Hamiltonian cycle will be present in any given topology, although in many topologies there is. When used as a *p*-cycle, a Hamiltonian will protect all spans in the network while using spare capacity on a minimum number of spans. Any other fully-restorable *p*-cycle solution that was non-Hamiltonian would need to have spare capacity present on a greater number of spans. To illustrate the superior efficiency that a Hamiltonian can offer, consider a 20-node, 40-span network (such as that seen in Figure 4.2(a)). An individual Hamiltonian *p*-cycle in such a network will obviously cover N on-cycle spans and therefore $S-N$ straddlers, and could thus have a hop-based logical redundancy of $N/\left[N(1)+(S-N)(2)\right]$ (the distance-weighted redundancy will obviously depend on the exact cycle chosen). In our 20-node, 40-span example, this is the astoundingly low value of 33.3%!

In the following sections we consider Hamiltonian *p*-cycles in both the conventional capacitated framework as well as the flat-capacity (or homogeneous) scenario. In a homogeneous network, only two pairs of fibers are present on each span: one pair for working demands and the other pair for protection. It is generally assumed that both fiber pairs have an identical number of wavelength channels. This configuration is also known as a "four-fiber" network, and is the paradigm for both the generalized loop-back and orientable cycle double cover methods mentioned in Section 3.12.2. This "flat capacity" framework might be a simplifying assumption or could instead reflect future network equipment characteristics where a DWDM system might be able to provide an extremely high number of channels (λs) on each fiber at little or no cost premium over systems with lower channel counts. For example, this would mean (not the reality today, however) that lighting 160 channels on every span would not be much more expensive than simply lighting 160 channels on the most heavily-loaded span and then a combination of 32, 16, 8, 4, and 1 channel units (as needed) on all the others. This is perhaps conceivable for the most advanced future DWDM technology one might imagine. But nevertheless, it is still interesting from a research perspective to look at homogeneous networks anyway. Although protection switching could still conceivably be done at the individual wavelength level, a more likely and advantageous scenario for such a flat-capacity network would be where protection is implemented at the whole-fiber level. A single *p*-cycle formed out of pairs of fibers would protect the single working fiber pairs on each span. These can also be described as "dark fiber" *p*-cycles because prior to failure and the associated protection switching, the *p*-cycle fibers will simply be unenergized, dark glass. Such *p*-cycles could easily be implemented with fiber-switching cross-connects (or pairs of SSIUs and OADMs for use directly as *p*-cycle nodes). If switching is done in this way at the whole-fiber level, then all the wavelength channels implicitly stay connected without the need for conversion because they are simply being moved from one piece of glass to another.

4.2. Role of Hamiltonian *p*-Cycles in a Capacitated Design

Before we go into the homogeneous case, let us first consider the non-homogeneous (or capacitated) scenario. Normally in a network we will see an uneven distribution of working capacity on spans. This will arise prior to the spare capacity placement (SCP) from the routing of working demands over their shortest paths on the network graph, in the same way this would normally be done for a span-restorable mesh.

4.2.1. Conventional *p*-Cycle Design Model

Here we present the most basic *p*-cycle model (from [29],[40]) for spare capacity design (SCP) after working demands have already been routed and working capacity values on each span are known. This is valuable general background even apart from our focus on Hamiltonians in this chapter, and we use it now to illustrate some general properties of capacitated designs. When determining the input sets and parameters, we make use of a preprocessing program as described in general in Section 2.3.5. This program [100] and the AMPL implementation of the model (Appendix B.1) were previously developed by John Doucette, a colleague in the TRLabs Network Systems group. The model requires sets of spans and candidate cycles, along with costs and working capacities for each span and two span-to-cycle relationship-defining parameters. Here we set the cost (c_j) parameters to either 1, or to the Euclidean distance between nodes, as explained later. The $x_{i,p}$ value represents how many protection paths cycle *p* can offer to span *i* (0, 1, or 2, depending on their relationship), and the $\theta_{j,p}$ value indicates if, when building cycle *p*, we need to reserve any spare capacity on span *j*. The model can be specified as follows:

SETS

S	Set of spans, indexed by *i* (failed) or *j* (surviving).
P	Set of candidate *p*-cycles, indexed by *p*.

PARAMETERS

c_j	Cost of one unit of capacity on span *j*.
w_j	Working capacity on span *j*.
$x_{i,p}$	Equal to 2 if span *i* straddles *p*-cycle *p*, 1 if span *i* is on *p*-cycle *p*, 0 otherwise.
$\theta_{j,p}$	Equal to 1 if span *j* is on *p*-cycle *p*, 0 otherwise.

VARIABLES

s_j	Spare capacity on span *j*.
n_p	Number of unit-capacity copies of *p*-cycle *p*.

OBJECTIVE FUNCTION

$$\textbf{SCP:} \quad \text{Minimize: } \sum_{\forall j \in S} c_j \cdot s_j \tag{4.1}$$

(Minimize spare capacity cost.)

CONSTRAINTS

$$\sum_{\forall p \in P} \theta_{j,p} \cdot n_p = s_j \qquad \forall j \in \boldsymbol{S} \tag{4.2}$$

(Ensures enough spare capacity is placed on every span to build all chosen *p*-cycles.)

$$\sum_{\forall p \in P} x_{i,p} \cdot n_p \geq w_i \qquad \forall i \in \boldsymbol{S} \tag{4.3}$$

(Ensures *p*-cycles chosen are adequate to protect all working capacity on every span.)

All variables are also restricted to taking non-negative integer values. The objective function shown in Equation (4.1) has the effect of minimizing the total spare capacity cost for the network design, while the two constraints in Equations (4.2) and (4.3) assert that the design selected must be equipped with enough spare capacity and provide an adequate level of protection for the working capacity on each span. Technically, of course, this is an NP-hard model, but in practice is often quite easy to solve quickly. For large networks, though, we may still need to limit the cycle set based on a metric as described in Section 3.8.1.

4.2.2. Hop-Based Solutions

To start with, we examine the solution to this model for both small and mid-sized networks. Figure 4.1(a) depicts the 13-node, 23-span "Canada" test network, with $\overline{d} = 3.54$. (We are not certain of the origin of this network but we received it from Oliver Yang's group at the University of Ottawa and later observed it in [101] as well.) Here we set all span costs (c_j values) to be 1. This network is Hamiltonian and for this particular test was given a demand matrix of one unit between every node pair. The working capacities (w_i values) on each span are indicated in Figure 4.1(a) and are quite variable with values over a range of between 1 and 14 working channels per span (158 in total). These are obtained through standard least-hop routing of all demand units between their

O-D pairs. The optimal design process has a universe of 410 distinct candidate cycles to consider and took 0.63 seconds to solve to optimality on "apollo."

We can see seven individual *p*-cycles in the solution and five unique cycle patterns. Three of these five patterns are Hamiltonian, while two are not. In Figure 4.1(b)-(f) the cycles chosen are illustrated with notations to indicate if they are Hamiltonian as well as how many unit-capacity copies of the pattern are selected (x1, x2, etc.). This design has a logical (hop-based) redundancy of $85/158 = 53.8\%$. To illustrate our general point that a single Hamiltonian will be more expensive (in a capacitated scenario) than an optimal design, we can first appreciate that the least-cost Hamiltonian would need seven units of capacity assuming the most highly loaded span (with $w_i = 14$) was configured as a straddler. In that case, it would need $13(7) = 91$ units of working capacity for a logical redundancy of $91/158 = 57.6\%$. This is higher than the optimal design, and although there is not a large difference in the redundancies, this simple example nevertheless

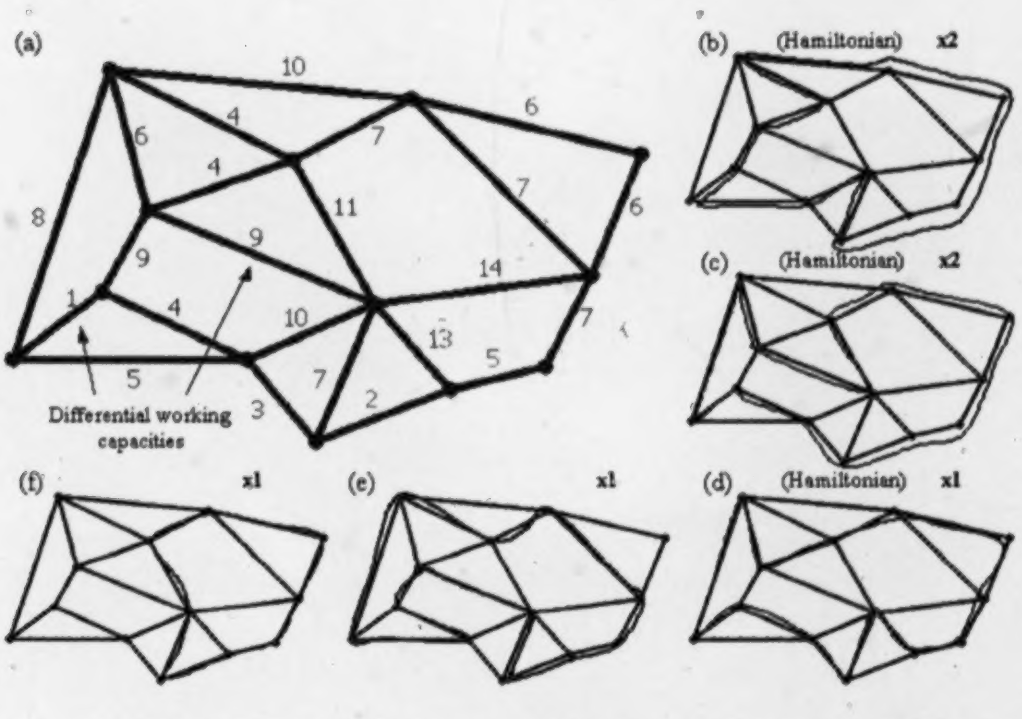

Figure 4.1. Optimal solution to "Canada" network (a) illustrating that in the capacitated design, some *p*-cycles may be Hamiltonian (b)-(d) but others may not (e)-(f) (from [14] (p. 721), [70]).

successfully illustrates the need (in general) for a solution with a mixture of Hamiltonian and non-Hamiltonian cycles.

To bring out the point further, we consider the larger (and also Hamiltonian) "20n40s1" test network (identical to "Net-A" in [87] (p. 11)) (with 20 nodes, 40 spans, and $\overline{d} = 4.00$), under the same parameters as before (all $c_j = 1$, a single unit of demand

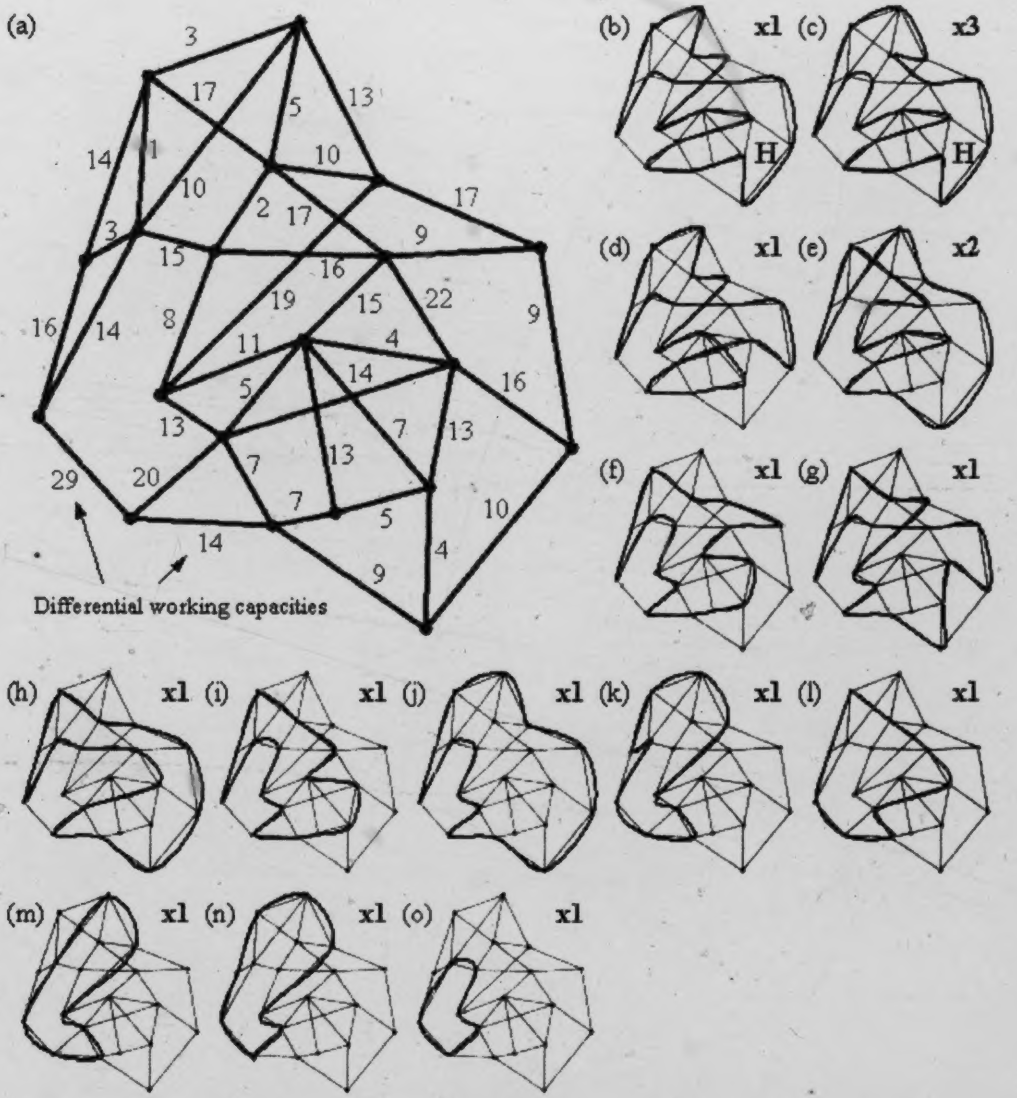
(a)
3
17
5
13
14
1
10
10
17
2
17
3
15
9
16
19
15
22
9
16
14
8
11
4
14
5
13
16
13
7
13
29
20
7
13
7
5
10
14
4
9
Differential working capacities
(b)
x1
H
(c)
x3
H
(d)
x1
(e)
x2
(f)
x1
(g)
x1
(h)
x1
(i)
x1
(j)
x1
(k)
x1
(l)
x1
(m)
x1
(n)
x1
(o)
x1

Figure 4.2. **Optimal solution to "20n40s1" network (a) again illustrating that a capacitated design may contain both Hamiltonian (b)-(c) and non-Hamiltonian (d)-(o) *p*-cycles.**

between every O-D pair, and least-hop working routing). Individual working capacities on spans range from 1 to 29 units with a total count of 456. This graph (depicted in Figure 4.2(a)) contains 59,904 possible cycles, *all of which* were offered to the solver. As expected, this design took considerably longer to solve (2336 seconds), now using the even faster server "liza." The solution in this case was within 0.27% of optimality.

A total of 254 spare channels were required, giving a logical redundancy of $254/456 = 55.7\%$. The best possible Hamiltonian with a capacity of 15 units per span (to protect the single span with the highest w_i value of 29 as a straddler) would thus have a logical redundancy of $300/456 = 65.8\%$. Note that this assumes the Hamiltonian cycle could be successfully placed to accommodate *all* spans with $w_i > 15$ as straddlers. If this were not possible, then the cost for a Hamiltonian solution would only increase. In contrast, the more economical multi-cycle answer has 17 individual *p*-cycles and 14 cycle patterns (shown in Figure 4.2(b)-(o)). Interestingly, only two of these patterns and four of the unit-capacity *p*-cycles were Hamiltonian in the overall design. Again, this shows (perhaps even more convincingly than the previous network) the need for a solution with a variety of cycles in order to reduce network cost as much as possible, not a single Hamiltonian *p*-cycle.

4.2.3. Distance-Based Solutions

In this section we look at the same two networks, but with distance-weighted capacity cost values. This is where the c_j parameters are proportional to the length of the span in the network diagram. The demand matrices, cycle sets, and solution tools stated previously are unchanged. For the Canada network, where least-hop routing now leads to w_i values between 1 and 13 with a total working capacity of 168 units, there were 13 unit-capacity *p*-cycles constructed out of seven distinct cycle patterns selected (in a solution that was strictly optimal). Only two of the individual *p*-cycles were Hamiltonians. The distance-weighted redundancy was calculated to be 65.9%. For the 20n40s1 graph, the working capacities were distributed between 1 and 34 for a total of 482 working channels. The solution in this case was within 0.14% of optimal, and here we can see 18 unique *p*-cycles made with 14 cycle patterns. An interesting observation is that *none* of these were Hamiltonian! This is perhaps to be expected, because in the distance-weighted framework, a slightly smaller cycle with many long straddlers (by distance) might have a lower individual redundancy than a poorly chosen Hamiltonian where the long spans are on the cycle. A distance-weighted redundancy of 60.1% was calculated, which is substantially lower than the 96.5% distance-weighted redundancy for the best single-Hamiltonian solution in the same network taken from the results presented in the next section.

4.2.4. Network Family Solutions and Hamiltonian Comparison

Here we compare the (distance-weighted) capacity cost for single-Hamiltonian versus multi-cycle solutions again, but this time over a range of network average nodal degree. This is a technique that has been used in previous work in the field (for example [35]) to see the effects of different techniques as network connectivity varies. Here, as connectivity dropped, we observed that past a certain point, the network became non-Hamiltonian so single-cycle solutions were not possible. For these cases we instead obtained the best two-cycle solution (so as to avoid infeasibility). The basic *p*-cycle SCP model was used to obtain the baseline results, and a slight variant was used to get the results with a cycle pattern restriction, as follows:

$$n_p \le \Delta \cdot {}^{bin}n_p \qquad \forall p \in \boldsymbol{P} \qquad (4.4)$$

$$\sum_{\forall p \in \boldsymbol{P}} {}^{bin}n_p = \mu \qquad (4.5)$$

The first constraint calculates a new set of variables ${}^{bin}n_p$ from the existing collection of n_p. The parameter Δ is a large positive constant (10000), and each ${}^{bin}n_p$ is restricted to taking on binary values of 1 or 0 to represent whether or not a certain *p*-cycle *pattern* is used, irrespective of the number of individual copies that are chosen. The second constraint forces the number of unique cycle patterns used to be exactly μ, a parameter that we set to be either 1 or 2 to obtain the data points of the following graph. We can thus see that these two constraints put together have the effect of forcing a specified number of cycle patterns while still leaving the solver the flexibility to provision enough unit-capacity copies to assure full restorability.

We began with the same 20-node, 40-span network seen in Figure 4.2(a) (and the inset in Figure 4.3) and progressively removed one span at a time in a pseudo-random way to create a family of networks, each with one fewer span than before. (This is the same method used in [35].) The same demand matrix with 1 unit between every O-D pair was used again as well. These were all solved using the "liza" CPU server, and were all within 0.5% of optimal for the baseline *p*-cycle designs. The Hamiltonian results were all optimal and the two-cycle solutions were strictly within 0.1%, although many of these were optimal as well.

In Figure 4.3, the significant cost premium for one- or two-cycle solutions can be clearly seen relative to efficient multi-cycle designs. On average, the Hamiltonian cycle chosen was 52.9% more expensive than the corresponding standard design, which can incorporate a variety of different *p*-cycles chosen to complement each other in an optimal way. Keep in mind that this Hamiltonian is the *best* one possible and other longer (by distance) Hamiltonian cycles may be even worse. The two-cycle solutions in the sparser, non-Hamiltonian graphs were more reasonable and were on average only 17.1% more

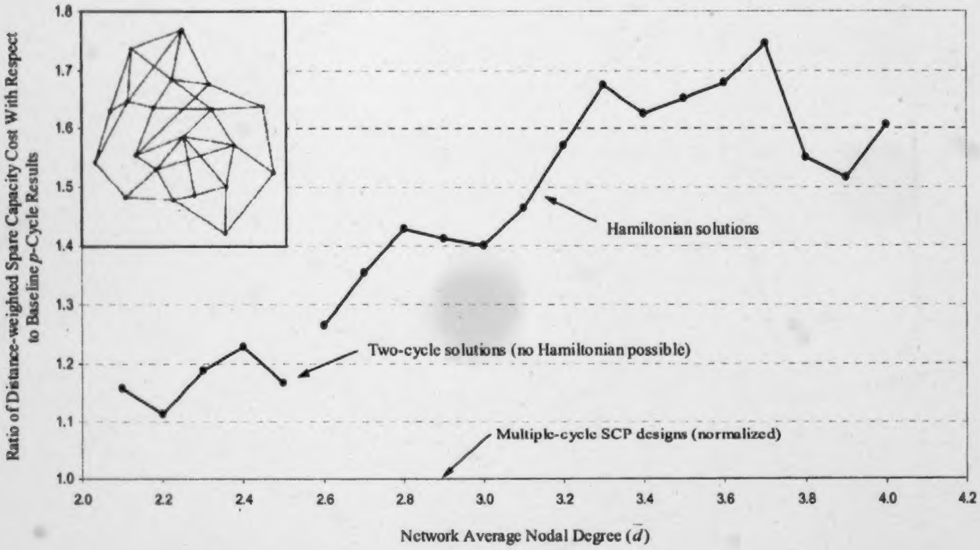

Figure 4.3. **Single Hamiltonian (or two-cycle) solutions can use significantly more capacity than a standard multiple-cycle design (from [70]).**

expensive than the baseline. We can observe from this graph that in general, the relative performance of the Hamiltonian solutions deteriorates as the network becomes more richly connected. This can be explained by observing that in the higher-degree networks, the optimal solver will have more cycle choices available to it and will thus be able to find lower-cost solutions. The percentage difference compared to the Hamiltonian solutions will then increase.

4.3. Hamiltonian *p*-Cycles in a Flat-Capacity Design

Let us now consider the case where the working capacity on every span is equal. "Equal" can mean two things – a single indivisible module on every span (the "unitary" case) or a collection of an equal number of individually managed channels on every span (the "flat divisible" case). In this section we consider the unitary case. (We also make the assumption that the network graph is Hamiltonian.) In this scenario it may seem intuitive that the optimal solution should be a single Hamiltonian because a collection of smaller cycles would seem to be gratuitously wasting capacity. This conclusion is actually correct[2]. The Hamiltonian will have N hops and thus, because it "touches" every node in the network, it will be able to offer protection to every span, in either the on-cycle or straddling sense. If the cost metric in the network is hop-based, then all Hamiltonian cycles are equally good. If weighted by distance, then the one with the shortest length will be the best compared to all the other candidates.

4.3.1. Hamiltonian Illustration and Calculation of Redundancy Lower Bounds

An example of this is shown in Figure 4.4, where in (a), the network with unit working capacities is depicted with optimal solutions for both the hop-based (b) and distance-based cases (c). These were solved with the standard *p*-cycle SCP model that allows multiple cycles to be chosen, but the actual result shows (as predicted) that both results consist only of single Hamiltonian cycles. In Figure 4.4(b) (hop-based), the cycle chosen is equivalent in cost to all other Hamiltonians (and has a logical redundancy of $13/23 = 56.5\%$), while in (c) we know the cycle chosen was the shortest Hamiltonian

---

[2] During revisions to this thesis, prepublication discussions of ongoing work by others make it clear that this is not the case when considering "flat divisible" capacity.

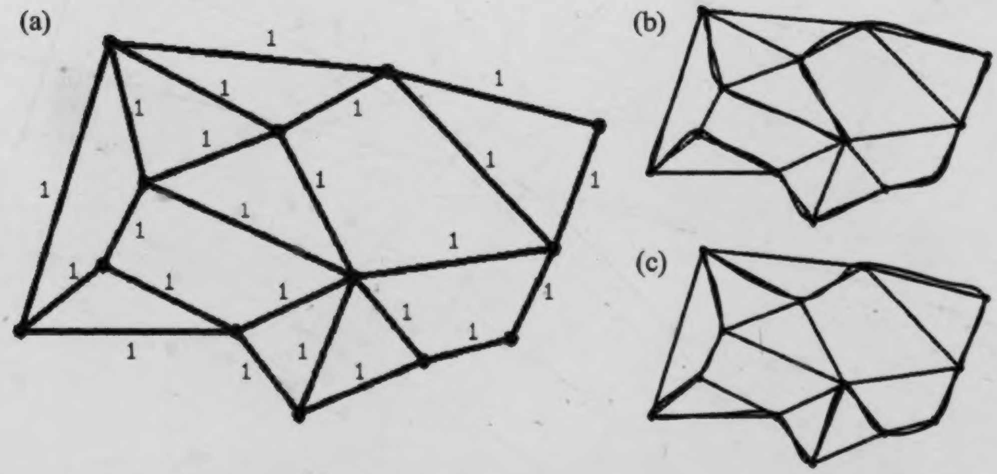

Figure 4.4. In a homogeneous, Hamiltonian network (a), a single *p*-cycle is optimal under both a hop-based (b) and distance-based (c) framework (adapted from [14] (p. 723),[70]).

available (now with a distance-weighted redundancy of 47.1%). We can observe that in the hop-counted case, the network redundancy figure of 56.5% is exactly as predicted by the $2/\bar{d}$ lower bound, presented next.

Based on the hypothesis that the optimal solution in a homogeneous (and Hamiltonian) network is always a Hamiltonian cycle, we can calculate a lower bound on redundancy for this particular type of network. If we say the network has N nodes, S spans, and average nodal degree $\bar{d} = 2S/N$ (which is the standard formula for calculating average nodal degree), the logical redundancy would thus be

$$\Re = \sum_{\forall j \in S} s_j \Big/ \sum_{\forall j \in S} w_j = \frac{N}{S} = \frac{N}{N\bar{d}/2} = 2/\bar{d}\,. \tag{4.6}$$

We describe this as a lower bound because it applies, of course, only to the optimal solution consisting of a single Hamiltonian. Other acceptable (but sub-optimal) *p*-cycle solutions could still have higher redundancy values, however.

Now taking a slight diversion, it is well known that the lower bound on the logical redundancy of a span-restorable mesh network is $\Re = 1/(\bar{d}-1)$ (as explained in [35]). A short paper on the theoretical underpinnings of *p*-cycles [56] was able to show that *p*-

cycles also exhibit the same lower bound, but the practical significance and way to fully realize this lower bound did not seem to be fully understood. We can now appreciate how *p*-cycles can actually "get there" when we recognize that homogeneous networks as just described fail to recognize or exploit a very crucial aspect of *p*-cycles, which is of course their ability to protect straddling spans with two units of working capacity on each one. Thus, starting from one of our Hamiltonian solutions, such as those pictured in Figure 4.4 for an example network, we could "load up" each of the straddling spans with one extra unit of capacity, for "free," with no changes required to the *p*-cycle to retain full restorability. Incorporating this slight twist into our redundancy calculations, we now have the ability to protect

$$\sum_{\forall j \in S} w_j = N + 2(S - N) = N + 2\left(\frac{N\bar{d}}{2} - N\right) = N(\bar{d} - 1) \tag{4.7}$$

working channels, thus making the overall redundancy

$$\Re = \sum_{\forall j \in S} s_j \Big/ \sum_{\forall j \in S} w_j = \frac{N}{N(\bar{d}-1)} = \frac{1}{(\bar{d}-1)}. \tag{4.8}$$

This result is recognizable as the well-known lower bound for both span-restorable mesh networks and *p*-cycle networks in theory, both mentioned above. However, this very efficient and desirable lower bound can now be actually associated with a constructible way to design a network that can easily reach the bound with a very simple provisioning technique. We will return to this idea later in the chapter.

The hop-based redundancy of 53.8% from Section 4.2.2 (for the same network as depicted in Figure 4.4) can be positioned mid-way between these two lower bounds. On one hand, it is below the $2/\bar{d}$ figure of 56.5%. This is because in some cases, the optimal multiple-cycle design may be able to use straddling spans to protect a full two units of working capacity, which the strictly homogeneous design can never do. But on the other hand, it makes sense that the capacitated-framework design will probably not be able to associate two units of working capacity with *every* straddler, and so its redundancy will then remain above the absolute lower bound of $1/(\bar{d}-1)$, which is

39.4% for this particular network. We can see advantages here again with the general capacitated design from its ability to exploit the full protection capability of *p*-cycles when needed to produce an efficient global design.

4.3.2. The *AE* Metric for Hamiltonians

As mentioned in Chapter 3, a metric called the *a priori* efficiency (*AE*) can be calculated for any candidate cycle. This is accomplished with the following formula, presented in [70],[71]:

$$AE(p) \equiv \sum_{\forall i \in S} x_{i,p} \Big/ \sum_{\forall j \in S} \theta_{j,p} \cdot c_j \,, \tag{4.9}$$

where $x_{i,p}$ is the number of protection paths provided by *p*-cycle *p* to span *i*, $\theta_{j,p}$ indicates whether *p*-cycle *p* crosses span *j*, and c_j is the cost of a channel on span *j*. The term "*a priori* efficiency" is used because we do not know in advance if an appropriate amount or distribution of working capacity will be present in order to allow us to actually *realize* the predetermined *AE* value. (This distinction was also made in Section 3.8.2.)

In general, the *AE* value will improve as cycles become larger and more straddling spans are available to be protected. It is tempting and somewhat natural to surmise that because a Hamiltonian is the largest cycle possible, that it must also have the highest *AE*. This assumption, if true, would open the door to a trivial search procedure for the most efficient *p*-cycle solution (in a flat-capacity scenario) – simply sort the cycle set by the *AE* metric, pick the one with the highest value, and a Hamiltonian would be guaranteed. A much more profound implication arises when we consider that the DCPC protocol ([40],[51]) is a *polynomial time* procedure for finding the maximum *AE* *p*-cycle (with distributed, statelet-based methods). If Hamiltonians always had the highest *AE*, then the DCPC protocol would be a way of finding them (which is known to be an NP-hard problem) in polynomial time!

Unfortunately (but quite interestingly), we can demonstrate that the connection between Hamiltonians and high *AE* values is not always true, at least in a logical hop-count scenario. To appreciate this, consider the elementary, contrived example in Figure

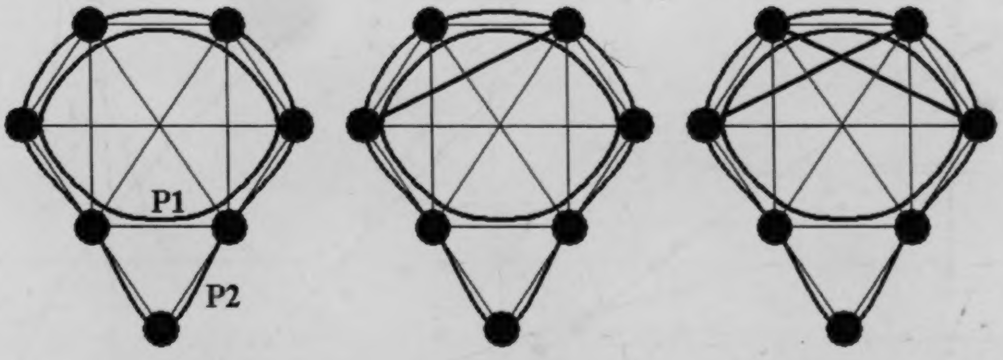

Figure 4.5. Example to illustrate a non-Hamiltonian *p*-cycle (P1) having (a) a lower *AE* value, (b) an equal *AE* value, and (c) a higher *AE* value than a larger, Hamiltonian cycle (P2) (from [14] (p. 725),[70]).

4.5. Here we see a Hamiltonian *p*-cycle, denoted P2, and a smaller, non-Hamiltonian cycle labeled P1. We can let this smaller cycle be just one hop shorter than the Hamiltonian, forcing it to exclude only one node from the route it takes through the network. If we want the two cycles to be as close as possible (which we do), then the best-case scenario will be that the excluded node is one with a degree of only 2. Thus, both spans incident on the excluded degree-2 node will not enjoy protection from the non-Hamiltonian cycle. Because it is one hop shorter than the Hamiltonian, cycle P1 will have one fewer on-cycle span, as well as one fewer straddling span because a total of two spans can no longer be protected. Thus, we can see that for the Hamiltonian P2, with N on-cycle spans and $S-N$ straddlers, the *AE* must be $\left[N+2(S-N)\right]/N=(2S-N)/N$. With $N-1$ on-cycle spans and $S-N-1$ straddlers, an *AE* value of $\left[(N-1)+2(S-N-1)\right]/(N-1)=(2S-N-3)/(N-1)$ can be calculated for the non-Hamiltonian P1. We can now compare these and assert that the Hamiltonian cycle will have a greater *AE* metric value *only* when:

$$\frac{(2S-N)}{N}>\frac{(2S-N-3)}{(N-1)}. \tag{4.10}$$

To convert this to a simpler and more directly applicable form, we note that the common formula for the average nodal degree, $\overline{d}=2S/N$, can be rearranged as

$S = N\bar{d}/2$. Substituting this into Equation (4.10) and simplifying, we are left with the very compact expression of $\bar{d} < 4$. This indicates, then, that for $\bar{d} \geq 4$ there may exist a cycle with an *AE* value equal to or greater than that of a Hamiltonian.

Returning now to Figure 4.5, we can see an example of this. In (a), the network has 13 spans, $\bar{d} = 3.71$, and the Hamiltonian *AE* is higher (2.71 vs. 2.67). By adding an extra span in (b) to make $S = 14$, the value of $\bar{d}$ increases to exactly 4.00. Here the *AE* metric for both cycles is equal with a value of 3.00 for each. Then in (c), another span is added for a total of 15 spans, and $\bar{d} = 4.29$. Here, the non-Hamiltonian cycle P1 wins out, with an *AE* of 3.33 in comparison to 3.29 for cycle P2.

While this may seem like a somewhat subtle point, its effect and consequences on possible Hamiltonian search strategies based on the *AE* metric cannot be simply overlooked. It is also important to note that even though a smaller cycle may have a superior individual *AE*, the "network" *AE* will be highest with a Hamiltonian because the smaller cycle will require a second complementary cycle to achieve full coverage.

4.4. Flat-Capacity Design in a Non-Hamiltonian Network

All of the above discussion on Hamiltonian *p*-cycles is based on the assumption that the network in question does, in fact, contain a Hamiltonian cycle. This is not always guaranteed, of course. A plausible assumption might then be that two cycles will be required for full protection. Again, this will be true in many cases, but unfortunately, it can be shown that for a non-Hamiltonian network graph, any arbitrary number of *p*-cycles may be required, depending on the specific topology.

Inspired by the idea of chains in "meta-mesh" design [102], we can propose a certain construct that, when added to a network, can easily force it to be non-Hamiltonian. For any chain of degree-2 nodes, we know that the chain must be traversed by a *p*-cycle in the on-cycle sense, because from the definition of a chain of degree-2 nodes, it is easy to see that the spans involved will never be accessible to any cycle in the straddling sense. Because we do not (normally) allow *p*-cycles to loop-back and "touch" a node more than once (unless they are non-simple cycles as in [82]), this imposes a limit of two chains per node if only one *p*-cycle is used for protection. Thus, to force more than one *p*-cycle, we simply need to increase the number of degree-2 chains incident on a common (or "anchor") node. When three or more chains are incident, at least two *p*-cycles will be required. When five or more chains are incident, at least three *p*-cycles will be required.

This was tested in the networks shown in Figure 4.6 through optimal solutions. All spans had an equal cost of 1 and the network graphs were specially constructed to have the feature of chain subnetworks and an anchor node like we just described. For each of cases (a)-(c), three chains were connected to the node marked "A". The results showed

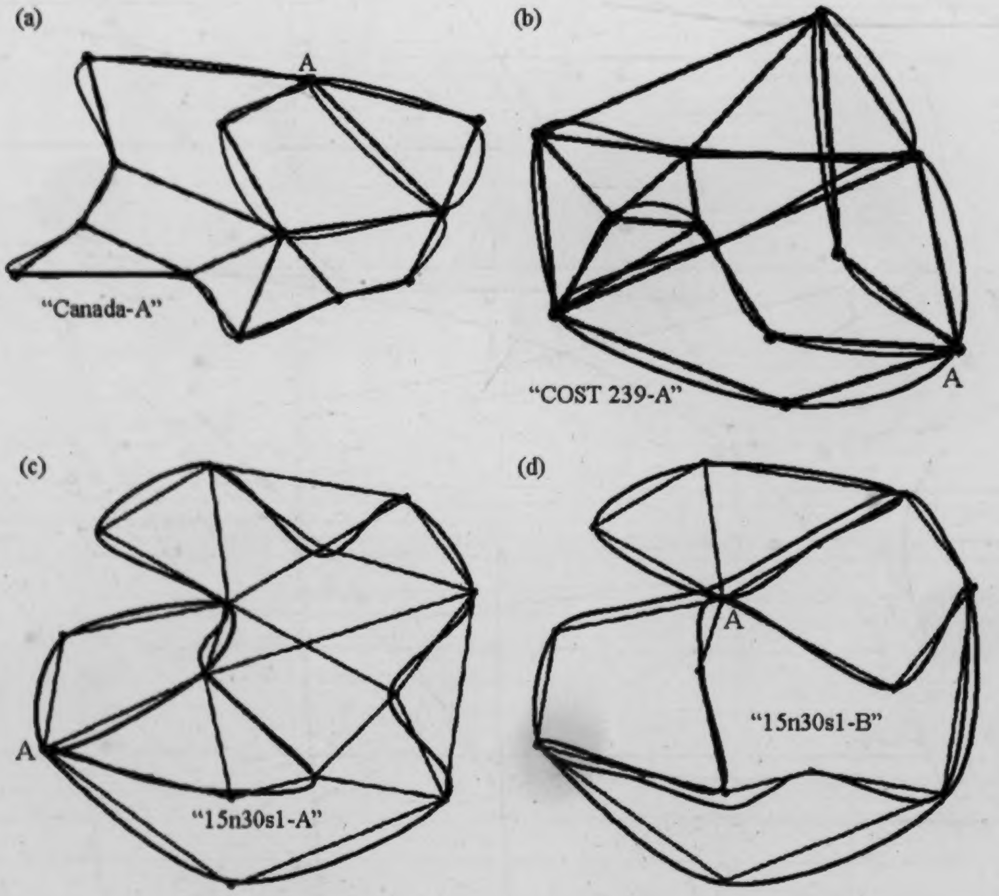

Figure 4.6. **Multiple-cycle designs required in non-Hamiltonian networks including those with three chains and two *p*-cycles (a)-(c), and one with five chains and three *p*-cycles (d) (adapted from [70]) (the graph in (a) is a variant of the Canada network seen earlier, while (b) is a variant of the well-known COST 239 graph as in [103]).**

that the strictly optimal solution (in these three cases) was the largest non-Hamiltonian possible (with $N-1$ nodes) coupled with the smallest *p*-cycle then required to achieve full protection coverage.

For the five-chain scenario in Figure 4.6(d), a three-cycle solution was returned by the solver, which was then verified by re-running the design with a constraint to force only a two-cycle solution (as in Section 4.2.4). The subsequent report of infeasibility from CPLEX was evidence that a three-cycle solution was actually required for the five-chain case, at least in this particular network. Thus, it is not possible to state with certainty that a two *p*-cycle solution will be required in the circumstances where a network is not Hamiltonian. In practice, most networks will not include our proposed construct with five chains converging on a single node, and thus may be amenable to protection with only two cycles, but this cannot always be guaranteed for any arbitrary network. Interestingly, the three *p*-cycles chosen for the scenario in Figure 4.6(d) are neither the largest nor smallest in the network, leaving us unable to offer a guideline for a design likely to be chosen, as we did in the two-cycle case where it seemed that the largest/smallest pairing was always or often the outcome. Here for the three-cycle scenario it seems that the cycles chosen will depend specifically on the exact network topology in question.

4.5. Semi-Homogeneous Networks Based on the *p*-Cycle Concept

Earlier in Section 4.3.1 we showed how the logical redundancy of a *p*-cycle network could exactly realize the theoretical lower limit of $1/(\overline{d}-1)$ in the case where all spans (both on-cycle and straddling) fully exploit the protection offered to them by a Hamiltonian *p*-cycle. We can position this as a special type of network, which we describe as "semi-homogeneous" to differentiate it from both the flat-capacity homogeneous case and the somewhat "random" fully capacitated designs. Figure 4.7 below shows how this would be realized in practice. In (a) we can see how the straddling spans can now double their working capacities to fully exploit the protection capabilities of the cycle and achieve the theoretical lower bound. In (b) we can see the "zoom-in" view of the on-cycle and straddling spans in the whole-fiber context, where part (c)

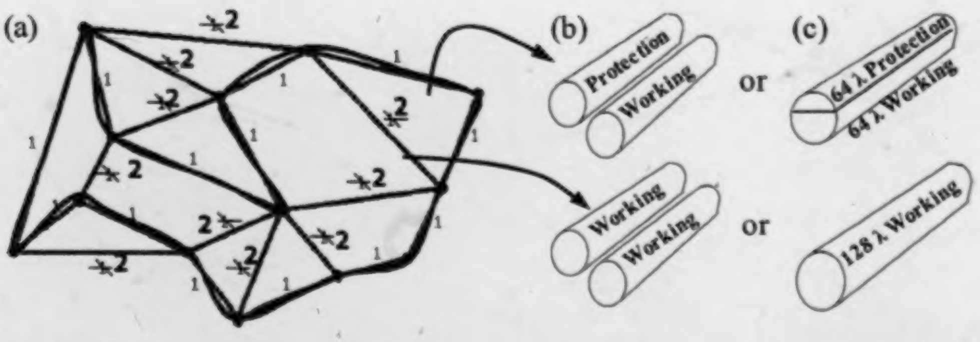

Figure 4.7. **A semi-homogeneous network with two capacity levels supported by a Hamiltonian *p*-cycle (a), along with the implications for each type of span in both the whole-fiber (b) and wavelength switching cases (c) (from [14] (p. 726),[70]).**

depicts the wavelength-switching context where the channels in a single fiber could be split up as required. Note that for this particular network the redundancy is at the (rather impressive) limiting value of 39.4%!

The reason we call this a semi-homogeneous network is because it does not relax the restrictions of homogeneity very much, and certainly does not allow arbitrary working capacities, but is still a departure from the strict flat-capacity definition of a perfectly homogeneous network. It is in a sense "in-between" these two extremes, so the semi-homogeneous descriptor seems to fit.

It is interesting to observe that although the working capacities can take on either of the two possible values permitted by semi-homogeneity (either one or two working fibers, depending on the orientation of the span with respect to the cycle), when coupled with the spare capacity required to build the *p*-cycle, the semi-homogeneous network actually turns out to be strictly homogeneous *in the total capacity sense*. Given that total capacity is what network operators actually build, this semi-homogenous paradigm (under the *p*-cycle framework) can make extremely efficient use of *total* flat capacity provisioned on each span. This design strategy could also be tailored so that spans with the highest expected growth would be arranged on the *p*-cycle as straddlers so they would use the first working fiber initially and then the second working fiber in the future as demand growth warranted, while still being protected by the single original *p*-cycle.

4.6. Summary

In this chapter we showed how Hamiltonian *p*-cycles are not always a better choice in a capacitated network graph, and also that a Hamiltonian might not have the best *AE* value in certain high-degree networks. We also proposed the idea of a semi-homogeneous network to fully realize the capacity benefits of *p*-cycles. The *p*-cycle SCP design model we presented in the process will be useful background in the next chapter, which looks at how *p*-cycles interact with other methods in a multi-method design.

Chapter 5. A Mixed Design Approach With *p*-Cycles

From prior work on ring-mesh hybrids (in [33], as discussed in Section 2.2.4), we can appreciate the general philosophical point that by using multiple protection and/or restoration methods in concert with each other (instead of a single pure architecture), a superior solution in terms of overall cost can be obtained. The conclusion to another recent paper also brings this point to the fore, and states, "We expect that in future real life networks there will be a mix of different resilience schemes, depending on a service differentiation" [90]. And although a number of studies have been done to examine the merits of *p*-cycles in comparison to other methods ([41],[89],[90],[91], among others that are not framed explicitly as comparison papers), there has been no work to date which considers how *p*-cycles might act in harmony with other schemes as part of a multi-method strategy. (Prior work on aspects of multi-*layer* design (for example [104],[105]) is distinct from our orientation of multi-method design in the *same* layer.) This chapter presents a framework for such multi-method design, developed as part of a collaborative project that was undertaken with François Blouin at Nortel Networks, and contains material that was first published at the DRCN 2003 conference [21].

5.1. Motivation

5.1.1. Differentiation of Service and Survivability Methods

In an optical network, not all services are necessarily equal in terms of the restoration times they require. A data backup to a remote location at 2 AM would be far more tolerant of delay and a brief interruption than a live video broadcast, for example. While all services present in a network could, of course, be protected with the fastest method available, this may not make sense economically, especially if faster protection is more expensive (as traditionally it is) and customers are only willing to pay for the precise grade of service they require. While it remains easier for a network planner to assume a single, all-encompassing grade of service and design the network accordingly, with the sheer size and cost of large optical networks fully considered, all opportunities to reduce cost must be exploited. Design methods that consider multiple grades of service are one such opportunity.

Two popular survivability methods in optical networks today (among others – see [25],[34]) are 1+1 APS and SBPP. SBPP is preferred because of its efficiency, but diverse 1+1 APS can offer a much faster protection switching speed. A combination of the two can be commonly seen, where SBPP protects most of the demand units to lower the overall capacity requirements but 1+1 APS is used to protect certain premium services. There can also be other methods present as well, such as existing legacy rings (which we ignored for this project). Unprotected or best-efforts services might also be found, along with mechanisms at different layers as previously mentioned (and further discussed in [25]). Because this study focuses on multi-method design, we neglect these inter-layer considerations, along with the unprotected services in the transport layer, simply because they require no protection capacity and are thus trivial to deal with. Similarly, introducing best-efforts services into our model and results to follow would only serve to unnecessarily confuse the central issues we are looking at.

One of these issues is the possible effect that *p*-cycles might have when added to a 1+1 APS and SBPP combination. We know from previous chapters that *p*-cycles have an array of attractive features, including capacity efficiency and restoration speed. Introducing them as a possible class of protection (CoP) to be used with a certain subset of demands might have capacity advantages that we hope to expose later in the study. It is somewhat intuitive why this might be the case when we reflect on the capacity and restoration speed details of each method as presented in Chapter 2 and Chapter 3. 1+1 APS will have (in general) the largest capacity requirement, *p*-cycles will likely be more efficient, and SBPP even better still. This is supported by the results in [63] as explained in Section 3.6.2. For the restoration times, our colleagues at Nortel Networks were able to provide estimates for all three methods considered, and these are presented in Table 5.1. For this table, the important issue is the relative ranking of the methods and that a difference between them does exist. We recognize, of course, that the actual restoration times could depend on product implementations and actual network details.

In this chapter, we rank all the methods according to their restoration times, and so when terms like "better," "worse," "higher," and "lower" are used, these are referring to the relative position of the methods in Table 5.1. We also talk about demand units

Class of Protection	Restoration time	*m*
1+1 APS	< 60 ms	1
p-Cycles	< 80 ms	2
SBPP	< 200 ms	3

Table 5.1. Typical restoration times for various protection and restoration methods considered (from [21]).

"moving down" or "jumping up" in the context of the quality of protection they enjoy, and this also refers to this restoration-time hierarchy.

5.1.2. Introducing *p*-Cycles to the Mix

With the CoP differentiation in mind, there may be a number of services (which generate demand units) that might require a faster restoration time than SBPP can provide. In a universe without *p*-cycles, 1+1 APS (out of the three alternatives in this study) would be the only suitable choice. However, some services might be able to easily tolerate the only slightly longer delay associated with *p*-cycles, and could thus use *p*-cycles as an alternative to 1+1 APS. We expect that with this expanded set of techniques to choose from, a service provider will be able to find a mapping of demand units to suitable protection methods that will ensure survivability, but at a lower cost compared to the suite of protection methods before *p*-cycles were introduced. We are especially interested, as mentioned, in the possible transition between 1+1 APS and *p*-cycles to see how many demand units can "bump down" to *p*-cycles which could still (in those cases) offer adequate restoration times to the carried services but now in a much more cost-effective way. This transition would have no impact on how customer requirements are met (some may only need this "better than SBPP" protection instead of a fully diverse, dedicated 1+1 APS arrangement), but may have the potential to unlock significant capacity currently invested in 1+1 APS paths for more efficient and flexible use when employed as a set of *p*-cycles.

5.1.3. Introducing Self Selection

After *p*-cycles have been included in the suite of methods, we can pose another interesting question. In the last section, although we were only considering a single network, the paradigm was still one where the design for each method would occur separately. The interesting issue in that case was the extent to which demands that considered 1+1 APS and *p*-cycles to be "time-equivalent" would "jump down" to *p*-cycle protection to save capacity. But now assuming that *p*-cycles are available, we can "flip" this question to ask: if a demand can be protected at an equal or lower cost with a better method (i.e. "cost-equivalent"), could it "jump up" to that higher CoP? (This may seem somewhat counter-intuitive, but as we will see later, a method that is more expensive in general may actually be a more cost-effective choice for a particular individual demand.) The key to doing this is to allow the design model to make the choice of method for each demand unit, subject to meeting the minimum requirement, of course. This is in contrast to the framework in the last section, which we refer to as the "assigned" case from now on, because the CoP for each demand (i.e. the minimum requirement) was simply given as an input to the optimization process. Now, in the "self-selected" case, the method chosen must still be adequate but can be an "upgrade," if this is possible without increasing the total capacity requirement. We are interested in looking at not only the possible capacity savings from this combined global optimization process, but also at the number of demands that might move up from one method to another to improve the service quality to customers at no additional cost to the carrier.

5.2. Mixed Multi-Method Design Model

To accomplish our objective, we now define a generic ILP model for the design of networks with all three methods just discussed: 1+1 APS, *p*-cycles, and SBPP. The total demand quantity between every node pair is now represented not as a single aggregate amount, but as the quantity of demand between each node pair that requires a certain method (or better). This model will be used to obtain results for both the assigned and self-selected cases, with only a slight modification (explained below). By doing this (as opposed to using three distinct models for the assigned case), we ensure the results are as comparable as possible while still respecting the study framework just set out.

5.2.1. Preliminary Details

The model to follow minimizes the total distance-weighted cost of both the working and spare capacity needed to provide 100% restoration for any single span failure. This optimization is performed in conjunction with a bi-criteria objective function (similar to [106], although here it has a different purpose). It asserts that when cost is equivalent, demands are to be "upgraded" to a better method if this can be done. The bi-criteria term thus has the effect of simply "biasing" or "nudging" the solutions (when capacity cost is the same) to the extreme that sees all demands being served with the best method possible. This helps us to provide more consistent comparisons, instead of comparing results where the allocation of demands to methods might be randomly assigned within the space of solutions available at a certain value of overall cost. This bi-criteria term has a surprising impact, as we will see later.

It is necessary to optimize working demand routing and capacity placement together with spare capacity design because the three methods we consider (and the specific ways we design solutions for them) may treat these two types of capacity differently and have varying requirements for both working and spare, making the only truly fair comparison one that compares everything.

For 1+1 APS, the demand units we provision are routed over both sides of the shortest cycle between end nodes. One side can be considered the working path and the other side as the protection path. Note that this is slightly different than a possible alternate method, which would involve using the shortest *route* for the working path and then the next shortest *disjoint route* for the protection path. In our case, because the cycle-finding tools were conveniently available, we adopted the cycle-oriented approach instead, which has the added advantage of selecting the lowest cost disjoint path *pair*, instead of possibly limiting the protection path choices as a result of blindly routing the working path first. Because both paths in an APS setup are carrying the signal at all times, we classify the entire shortest cycle as working capacity and make no distinction between the two sides. Thus, in our model, there is no spare capacity provisioned for a 1+1 APS path.

We look after the *p*-cycle design in a fairly standard way – the working demands that require *p*-cycle protection (treated as a group for each O-D pair) simply take the shortest routes, with working capacity being provisioned accordingly. Spare capacity is then provisioned (with standard methods as in [40]) to support the *p*-cycle set that is selected.

For SBPP, we again adopt a cycle-oriented approach. The collection of demand units between a certain O-D pair that use the SBPP method will be assigned to a disjoint path pair between their end nodes. (Note that a disjoint path pair is in one sense just another way of describing a cycle, and we actually use our cycle-finding tools to enumerate them.) But where, for 1+1 APS, we made no distinction in the optimization model between the working and protection paths on either side of a cycle (because it was unnecessary), with SBPP we must do so, because the primary and backup paths require different treatment. This is why we adopt the disjoint path pair terminology, because now, even though the two paths are still assigned (or not) as a pair, they need to be identified as either primary or backup and have capacity provisioned accordingly. The primary side will need dedicated capacity, but the backup side can share spare capacity with other backup paths where possible as long as the corresponding primary paths will never fail together and try to activate their backups at the same time. For this reason, we always use the shortest side of the path pair as the primary and the longer side as the backup (the dedicated capacity will be cheaper if we always put it on the shorter side).

For each O-D pair, a number of these shortest path pairs are provided to the solver as possibilities (in the same way a traditional mesh network design receives input sets of possible restoration routes). Note that in this study, we do not impose any limitation on the extent of possible spare capacity sharing for SBPP (as in [35]), other than the basic constraint requiring the primary paths to be disjoint so contention will not arise. Because of the path pair orientation, the SBPP portion of our model is slightly different than the prior model for spare-only SBPP design in [35] which considers backup paths as independent entities, but like the model in [35], we only choose a single SBPP path (pair, for us) for all the demand units associated with a particular set of O-D nodes.

It is important to note that this is *not* a joint optimization in the sense that there is any independent flexibility in the routing of working paths to optimize capacity. We are optimizing predetermined working capacity placement over a variety of protection and restoration methods, as opposed to optimizing the actual working paths and working capacity placement over a variety of working path choices, as might be done for a single method design. The only exception is in the SBPP case where the ability to affect the working and spare capacity is present to a certain degree due to the choice between available path pairs. Our use of the term "joint" is thus meant mostly to reflect that both working and spare capacity are included in the model, not that they are both optimized over an independent array of possibilities. For 1+1 APS and *p*-cycles, the working capacity is placed in a purely deterministic way, and for SBPP, although the primary paths are not placed in quite such a controlled fashion, their selection is strongly coupled with the selection of backup paths because the model selects the best SBPP path *pair*, which thus keeps the SBPP element of the optimization process distinct from a conventional joint optimization as we would usually describe it (in the case of a span-restorable mesh, say). To reduce complexity, we assume a fully VWP network with no wavelength blocking or wavelength conversion issues.

5.2.2. Multi-Method ILP Formulation

After this preliminary discussion, the full model can now be expressed below. Note that here we represent each method by its method number (*m*) as given in Table 5.1.

SETS

$\boldsymbol{M}$	Set of restoration methods, indexed by *m*.
$\boldsymbol{S}$	Set of spans, indexed by *i* (failed) or *j* (surviving).
$\boldsymbol{D}$	Set of demand relations, indexed by *r*.
$\boldsymbol{P}$	Set of candidate *p*-cycles, indexed by *p*.
$\boldsymbol{R}_r$	Set of candidate SBPP path pairs for *r*, indexed by *b*.

PARAMETERS

ε	A small positive constant (0.0001).
Δ	A large positive constant (100000).
c_j	Cost of one unit of capacity on span *j*.
$d^{m,r}$	Number of demand units for relation *r* that require method *m*.
t_j^r	Equal to 1 if span *j* is on the shortest cycle between the end nodes of relation *r*, 0 otherwise.

e_j^r	Equal to 1 if span *j* is on the shortest route between the end nodes of relation *r*, 0 otherwise.
$x_{i,p}$	Equal to 2 if span *i* straddles *p*-cycle *p*, 1 if span *i* is on *p*-cycle *p*, 0 otherwise.
$\theta_{j,p}$	Equal to 1 if span *j* is on *p*-cycle *p*, 0 otherwise.
$\delta_{j,b}^r$	Equal to 1 if span *j* is on the primary (shorter) side of SBPP path pair *b* for relation *r*, 0 otherwise.
$\phi_{j,b}^r$	Equal to 1 if span *j* is on the backup (longer) side of SBPP path pair *b* for relation *r*, 0 otherwise.

VARIABLES

w_j	Working capacity on span *j*.
w_j^m	Working capacity on span *j* for method *m*.
s_j	Spare capacity on span *j*.
s_j^m	Spare capacity on span *j* for method *m*.
$\alpha^{m,r}$	Number of units for relation *r* that use method *m* (equal to $d^{m,r}$ for assigned case).
n_p	Number of unit-capacity copies of *p*-cycle *p*.
q_b^r	Number of units for relation *r* that use SBPP path pair *b*.
v_b^r	Equal to 1 if relation *r* uses SBPP path pair *b*, 0 otherwise (1/0 variable).

OBJECTIVE FUNCTION

MMSCP: Minimize: $$\sum_{\forall j\in S} c_j\cdot\left(w_j+s_j\right)+\varepsilon\cdot\sum_{\forall m\in M}\sum_{\forall r\in D} m\cdot\alpha^{m,r} \qquad (5.1)$$

(Minimize capacity cost and use the best methods.)

CONSTRAINTS

$$\sum_{\forall m\in M} w_j^m = w_j \qquad \forall j\in S \qquad (5.2)$$

(For every span, the working capacity must equal the sum of working capacity for all methods.)

$$\sum_{\forall m \in \boldsymbol{M}} s_j^m = s_j \qquad \forall j \in \boldsymbol{S} \qquad (5.3)$$

(For every span, the spare capacity must equal the sum of spare capacity for all methods.)

$$\sum_{\forall m \in \boldsymbol{M}} \alpha^{m,r} = \sum_{\forall m \in \boldsymbol{M}} d^{m,r} \qquad \forall r \in \boldsymbol{D} \qquad (5.4)$$

(For every relation, the total quantity of demand assigned to all methods must equal the total quantity required.)

$$\sum_{\forall o \in \{1..m\}} \alpha^{o,r} \geq \sum_{\forall o \in \{1..m\}} d^{o,r} \qquad \forall m \in \boldsymbol{M}, \forall r \in \boldsymbol{D} \qquad (5.5)$$

(For every relation, the total demand is protected with an assigned method or better.)

$$\sum_{\forall b \in \boldsymbol{R}_r} q_b^r = \alpha^{3,r} \qquad \forall r \in \boldsymbol{D} \qquad (5.6)$$

(For SBPP working demands, use method 3 protection.)

$$\sum_{\forall b \in \boldsymbol{R}_r} v_b^r = 1 \qquad \forall r \in \boldsymbol{D} \qquad (5.7)$$

(For SBPP demands, only one path pair can be used.)

$$\Delta \cdot v_b^r \geq q_b^r \qquad \forall r \in \boldsymbol{D}, \forall b \in \boldsymbol{R}_r \qquad (5.8)$$

(For SBPP demands, only the backup path of the path pair selected in (5.7) can be used for restoration.)

$$\sum_{\forall r \in \boldsymbol{D}} t_j^r \cdot \alpha^{1,r} \leq w_j^1 \qquad \forall j \in \boldsymbol{S} \qquad (5.9)$$

(For APS demands, working capacity is asserted on all spans of the shortest cycle between O-D nodes.)

$$\sum_{\forall r \in \boldsymbol{D}} e_j^r \cdot \alpha^{2,r} \leq w_j^2 \qquad \forall j \in \boldsymbol{S} \qquad (5.10)$$

(For *p*-cycle demands, working capacity is asserted on all spans of the shortest route between O-D nodes.)

$$\sum_{\forall r \in \boldsymbol{D}} \sum_{\forall b \in \boldsymbol{R}_r} \delta_{j,b}^r \cdot q_b^r \leq w_j^3 \qquad \forall j \in \boldsymbol{S} \qquad (5.11)$$

(For SBPP demands, working capacity is asserted on the primary side of the chosen path pair between O-D nodes.)

$$\sum_{\forall p\in P} x_{i,p}\cdot n_p \ge w_i^2 \qquad \forall i \in S \quad (5.12)$$

(For each span failure, the set of *p*-cycles must be adequate to protect all *p*-cycle working capacity.)

$$\sum_{\forall p\in P} \theta_{j,p}\cdot n_p \le s_j^2 \qquad \forall j \in S \quad (5.13)$$

(For every span, there must be sufficient spare capacity to support the set of *p*-cycles used in (5.12).)

$$\sum_{\forall r\in D}\sum_{\forall b\in R_r} \delta_{i,b}^r\cdot\phi_{j,b}^r\cdot q_b^r \le s_j^3 \qquad \forall (i,j)\in S^2 \mid i\ne j \quad (5.14)$$

(For each span failure, there is enough spare capacity on surviving spans to support all activated SBPP backups.)

Except for the 1/0 variables indicated above, all other variables are limited to non-negative integer values. To employ this model in both cases (assigned and self-selected), we either use $\alpha^{m,r}$ as a variable (as specified above) to allow the ILP solver the flexibility to determine its value in the self-selected case, or to force the use of specific methods in the designs (the assigned case), we change $\alpha^{m,r}$ to a parameter, with values corresponding to the $d^{m,r}$ parameters that encode the minimum method requirements. The constraints in Equations (5.4) and (5.5) are thus redundant in the assigned case, although for practical simplicity we left them in the model. As might be apparent from the number of constraints and variables, in the self-selected case this can be quite a complex model. For all tests in this chapter, we again used our fastest CPU server ("liza"). The model above was implemented in AMPL (see Appendix B.2) and a custom data file preparation and preprocessing program was developed based on code modules (including [100]) available in the Network Systems group for other preprocessing tasks (by John Doucette and various co-op and/or summer students). However, our program did not use any candidate cycle preselection, and thus, solutions for larger networks than those seen in this study could likely be obtained if this preselection capability were implemented. On the networks we did use, though, some interesting results could be seen, which we explore in the following sections.

5.2.3. Test Networks

We chose both a sparser, "ladder"-type graph and a richly connected European network as test cases. The European network was the well-known COST 239 study topology as in [103] (with span distances from [58]). The "Generic U.S." network was supplied by colleagues at Nortel and is a scaled-down version of a more detailed U.S. network graph. A realistic demand matrix was also supplied for Generic U.S. and we make use of it later in Section 5.5. Both networks are shown below in Figure 5.1. In all our tests in this chapter, the span cost values for both networks were distance-based.

In the next two sections, however, we make use of some standard test demand patterns to gain more insight into our new model. A "flat" demand matrix of 20 units between each O-D pair was provisioned and subsequently split into variations with each CoP being allocated either 15%, 30%, or 55% of the total demand (corresponding to exactly 3, 6, or 11 units, respectively, out of the 20 units in total). By stepping through all the possible arrangements of these 15%-30%-55% patterns, we can see how our method behaves as each survivability method is responsible for a greater or lesser portion

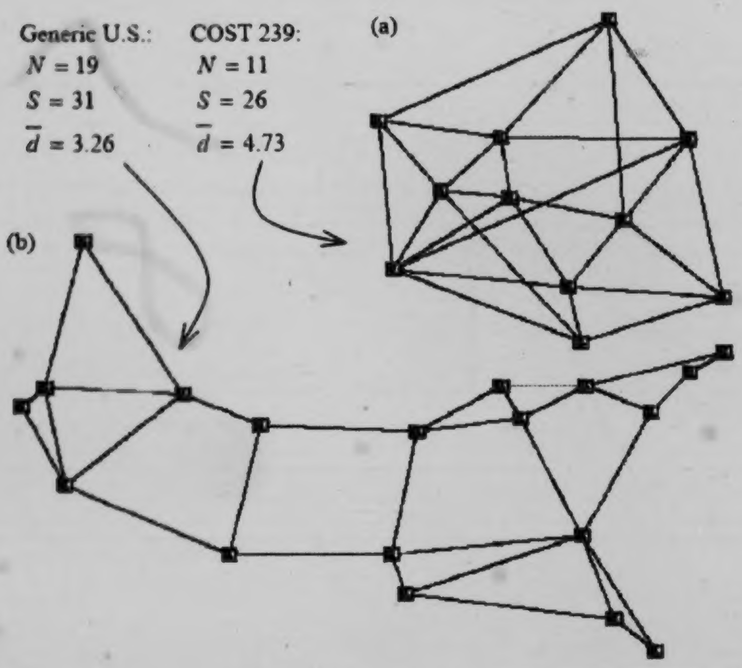

Figure 5.1. Test networks used: (a) COST 239 and (b) Generic U.S. (from [21]).

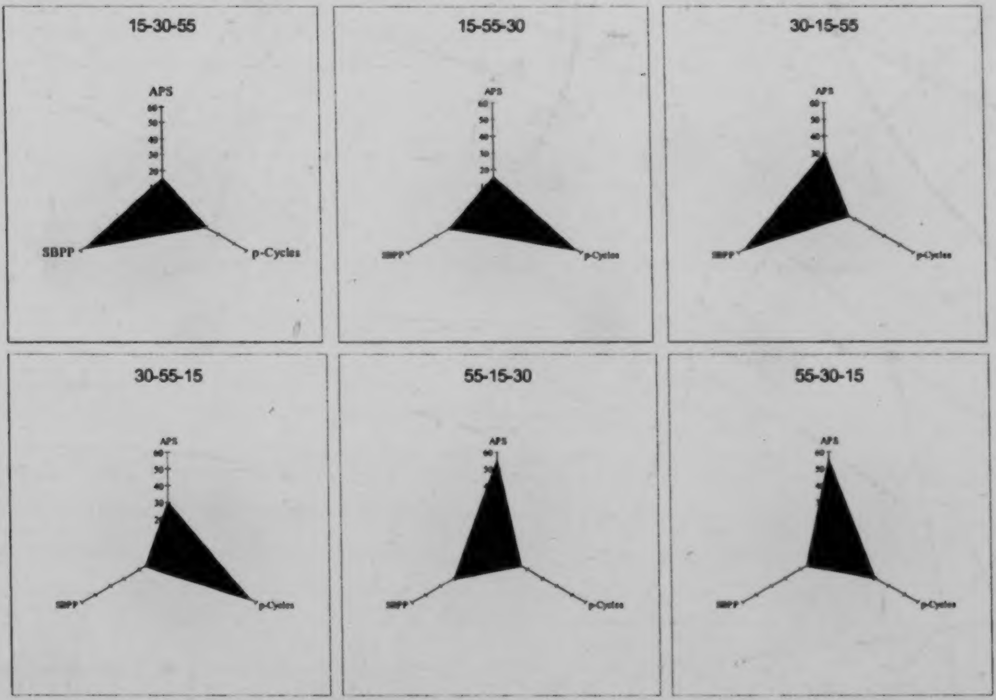

Figure 5.2. Illustration of different demand patterns specified for mixed service classes (in order of % APS - % *p*-cycles - % SBPP) ([107]).

of the total demand. This strategy of using multi-service demand mixes is inspired by a similar method used in [107]. Figure 5.2 illustrates what each demand pattern looks like when represented in a three-dimensional space, with one axis for each survivability method.

We provided at least 10 eligible SBPP path pairs between each O-D node pair for both test networks. Also in both cases, the full cycle set was made available to the ILP solver to use in building *p*-cycles. The Generic U.S. network had a total of 341 cycles while COST 239 had 3,531.

5.3. Results for Assigned Protection Designs

In this section we look at the effect of introducing *p*-cycles into the suite of methods, and make use of our model above, modified as previously described to fix the $\alpha^{m,r}$ terms as parameters. This means that $\alpha^{m,r}$ $(=d^{m,r})$ simply specifies how many demand units for each O-D relation (by our assignment) will use each survivability method for their protection or restoration. When specifying these as inputs, we of course assign each demand unit to its minimum required method. Note that in the case where we define *p*-cycles to be unavailable as an option, the minimum required method for certain demands would then need to be 1+1 APS instead. We would allocate all demand units only between 1+1 APS and SBPP in such a case, and set $d^{2,r}$ (and hence, automatically, $\alpha^{2,r}$) to zero. (The "2" is the numerical "method" descriptor for *p*-cycles as given in Table 5.1. To recap, 1+1 APS has $m = 1$, *p*-cycles are represented by $m = 2$, and $m = 3$ indicates SBPP.)

Our task is now to compare the total, distance-weighted capacity cost over the six demand mixes, with and without *p*-cycles as an available method. This will quantify the cost improvement from moving from 1+1 APS to *p*-cycles for those demands for which anything ≤ 80 ms is an adequate restoration time. In the figures below, we indicate the demand mix in a "15-30-55" format, where the first number is the percentage of demand units requiring 1+1 APS (60 ms), the second number is the percentage using *p*-cycles (80 ms), and the final number is the percentage that can tolerate SBPP (at 200 ms). However, this notation only specifies the minimum requirement, so when *p*-cycles are not available, the first and second two percentages will both be allocated to 1+1 APS (as explained above).

We arranged the "with" and "without" (*p*-cycles) bars on the graphs below in pairs, and so for each demand mix, both of these scenarios appear side-by-side for ease of comparison. Each bar shows the breakdown of the total capacity into the amounts required for each of the methods. The first three bars in each figure are for pure designs with each technique alone, simply to illustrate how the methods perform individually when no demand split is imposed. We can see that for Generic U.S., a reduction of 9.8% and 22.7% from the (most expensive) 1+1 APS solution can be obtained for *p*-cycles and SBPP, respectively. For COST 239, the corresponding reductions are 30.0% and 41.3%. COST 239 thus appears to be much more amenable to *p*-cycles (compared to Generic U.S.), at least when implemented as a pure architecture.

Figure 5.3 below shows the Generic U.S. results. These were all solved to within 0.01% of optimality. For the varying percentages of 1+1 APS demand that shifted to *p*-

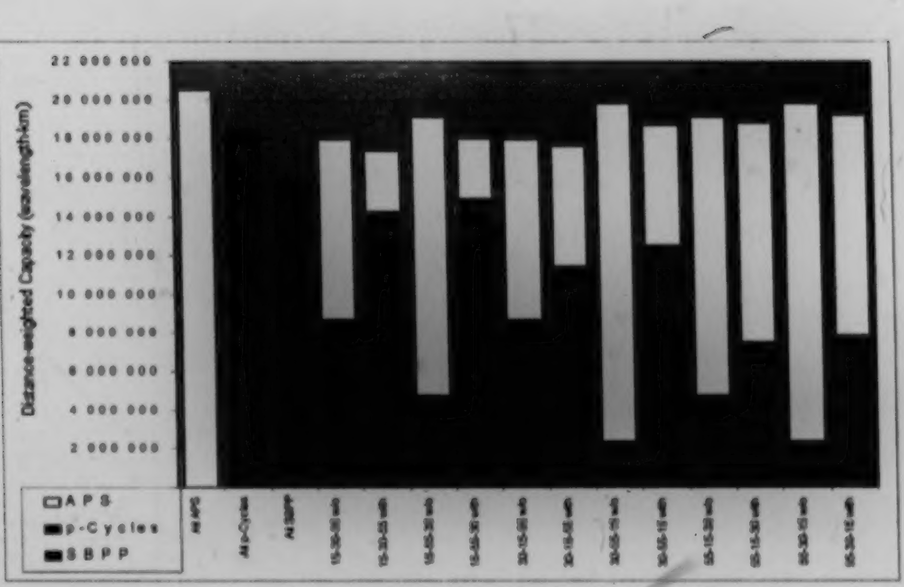

Figure 5.3. Capacity results using assigned model with Generic U.S. network (from [21]).

cycles (15%, 30%, or 55%), average improvements (respectively) of 1.6%, 3.2%, and 5.7% were seen.

Because of the scenario described earlier where *p*-cycle demands use 1+1 APS when *p*-cycles as a method are not available, several bars in Figures 5.3 and 5.5 are equivalent. For example, the bars for "15-30-55 w/o" and "30-15-55 w/o" are equivalent solutions because in both cases (due to the absence of *p*-cycles), the actual split is 45% to APS and 55% to SBPP. This can be seen in other "w/o" cases where the sum of APS and *p*-cycle demand is identical as well.

Initially we might have expected a set of larger reductions. (It can generally be appreciated that *p*-cycles may provide opportunities for spare capacity sharing whereas 1+1 APS cannot.) However, the issue is, "how many of them can we exploit?" We know from earlier chapters that to take full advantage of *p*-cycles, we need enough working flows to use (and share) the protection capacity. And of course, it is especially important to have enough working capacity on all the straddlers (otherwise the *p*-cycles will be limited in terms of efficiency). This particular topology (Generic U.S.) would seem to be not as amenable to *p*-cycles as some others, due to its "ladder"-type construction. A lot of the East-West demand routing in this continental U.S. graph will take place over the "sides" of the ladder (i.e. the on-cycle spans, for many of the possible *p*-cycles), and relatively few demands will tend to be routed on spans with a North-South direction (especially in the middle of the country). (In fact, sometimes routing demands through a rung of the ladder is impossible due to the routing infeasibility explained in [35], where the first path is routed in a way that precludes any other disjoint paths from even existing. A canonical example is shown in Figure 5.4, and this construct is generally known in survivable networking as a "trap topology" (as in [14] (p. 208),[108]).) But the rungs in the ladder will often be the straddling spans on many of the *p*-cycles, which are exactly where we need demand to be routed the most! For this reason, we speculate that other graphs, which are more densely connected (or geographically distributed in a more uniform style), may perform better in terms of the capacity reduction.

Now we can look at the results from COST 239. This network was more difficult to solve due to the larger number of cycle candidates and so these results were given a slightly more relaxed tolerance of 0.1% of optimal. But the high average nodal degree in this case is an auspicious sign because demands will now more frequently be able to cross straddlers as a part of their route. The average reductions for the 15%, 30%, and 55% demand shifts from APS to *p*-cycles are now 5.4%, 10.6%, and 18.1% in the total capacity cost. These are much better numbers than those for Generic U.S. and are notable because we are not even shifting *all* the demand to *p*-cycles, but only a portion, yet the reductions in capacity are still in the range of 5 to 18%. Figure 5.5 illustrates the results for COST 239 in the same format as before. In general then, to answer our original question, we see that adding *p*-cycles to the suite of methods can produce a significant savings, but that the benefits seem to depend on the topology of the network, with more densely connected networks seeing a more pronounced benefit.

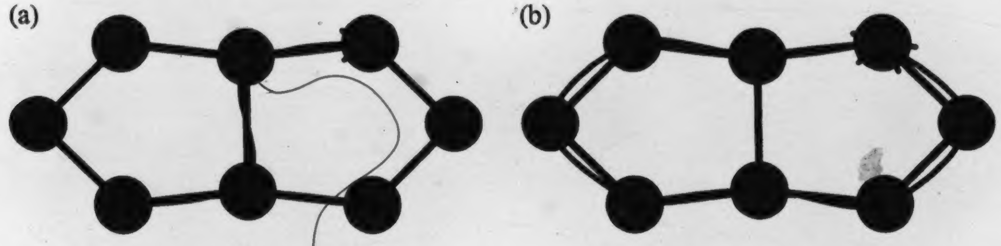

Figure 5.4. Illustration of (a) a routing infeasibility that can prevent a second disjoint path from existing, and (b) a solution with two paths both avoiding the middle "rung" span.

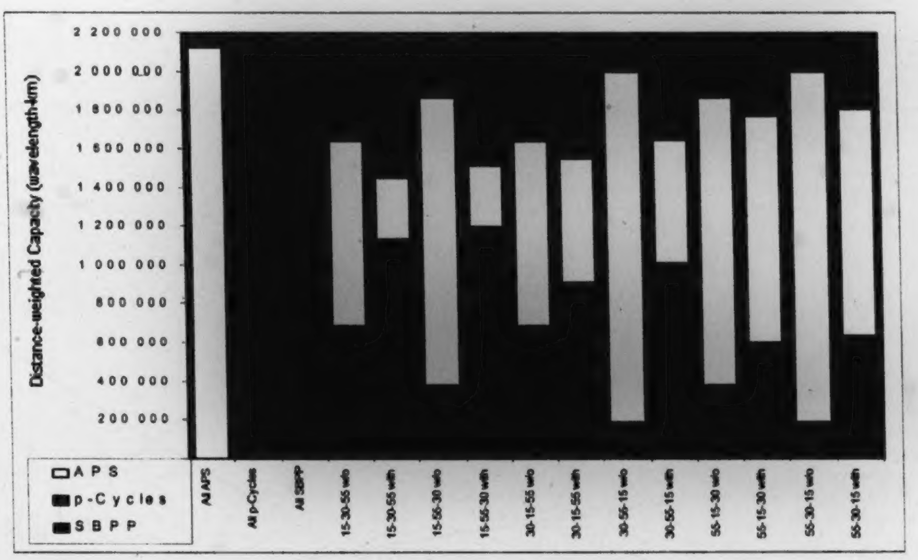

Figure 5.5. **Capacity results using assigned model with COST 239 network (from [21]).**

5.4. Results for Self-Selected Designs

This section is where we get the chance to test one of the main features of our new model, the ability to allow the solution model itself to select the highest quality method that satisfies each demand's requirements. This means demands may move up to a higher grade of service than they require. For example, *p*-cycles could move up to 1+1 APS, or SBPP could move up to either of the other two methods. The transition from SBPP to *p*-cycles would be especially advantageous, because the restoration time would be more than halved (from 200 ms to 80 ms). For this we make use of the model exactly as specified in Section 5.2.2, with the $\alpha^{m,r}$ values as decision variables.

In the previous section, we discussed how the cost was adversely affected by higher amounts of capacity needed in a *p*-cycle design when the demand routing tended (out of necessity) to favor the on-cycle spans as opposed to the straddlers. In such a case, the spare capacity we must provision is not really being used to its full potential. Our motivation in this part of the study thus arises quite naturally from the idea that the SBPP-grade demands could perhaps take advantage of the available but unused protection capability ("slack") in a *p*-cycle design (unless it were fully semi-homogeneous, in which case there would be no slack, but in the majority of cases this is highly unlikely). So when the solver determines that it is more efficient in terms of capacity (or equally efficient) to do so, demands will get "upgraded" to the highest level of service possible. (The bi-criteria objective term becomes responsible for the biasing when cost is exactly equal.) There is really no reason to avoid using such a strategy – a carrier could either offer the upgrades to its customers for free, or possibly exploit this ability to offer higher quality protection to increase its revenue. In any case, it will be interesting what the extent of the upgrades will be for both of our two test networks.

We again use the demand matrix with 20 units per O-D pair, along with the suite of 15%-30%-55% mixed patterns. The comparison in this section is between the self-selected approach (including *p*-cycles as an option), and the assigned approach including *p*-cycles from the previous section. To easily illustrate the changes, they are positioned on the charts in pairs (as done previously). This will let us see how the capacity is distributed between the three methods before and after we allow demands the freedom to upgrade. A bit later, we will also look at the profile of individual demand units in these same two cases to see the change in the number of demands relying on each method, as opposed to just the aggregate capacity totals.

Results are shown in Figures 5.6 and 5.7 for Generic U.S. and COST 239 respectively. The cost decrease was 4.8% (on average) in Generic U.S. and 2.2% (on average) in COST 239. This decrease was in relation to the assigned case with *p*-cycles. Compared to our starting position in the last section of an assigned approach *without p*-cycles, however, the cost reductions are 8.1% and 13.3% for Generic U.S. and COST 239 respectively.

There are some interesting things we can see in these two graphs. Notice the shifts in capacity from one method to another, especially when SBPP has a large portion of the demand to start with (the 55% cases). Not only does a lot of capacity shift to *p*-cycles, but there are shifts to 1+1 APS as well! Initially this was surprising and unexpected. After reflection though, there is a good reason why this happens, which we explain a bit later in the chapter.

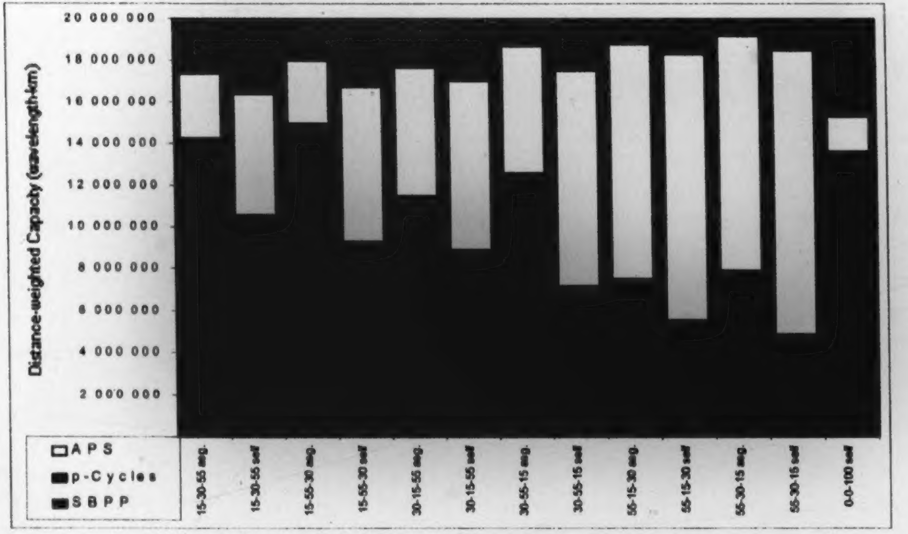

Figure 5.6. Capacity results using self-selected model with Generic U.S. network (from [21]).

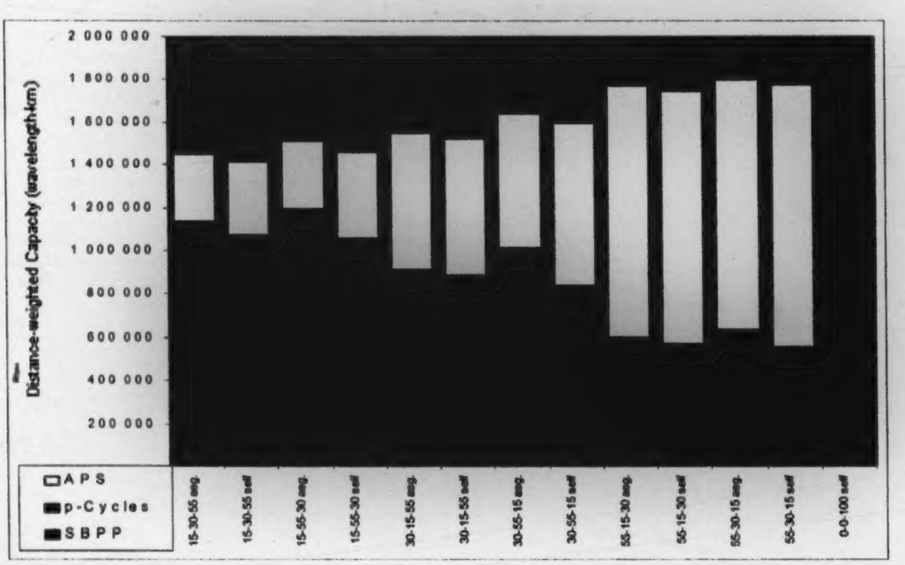

Figure 5.7. Capacity results using self-selected model with COST 239 network (from [21]).

The capacity totals do not clearly show how many individual lightpaths get the opportunity to upgrade to a better method. Out of the fixed quantity of demand units in each network (20 units per O-D pair), Figures 5.8 and 5.9 show the number of units protected by each method in the assigned and self-selected designs. These also show

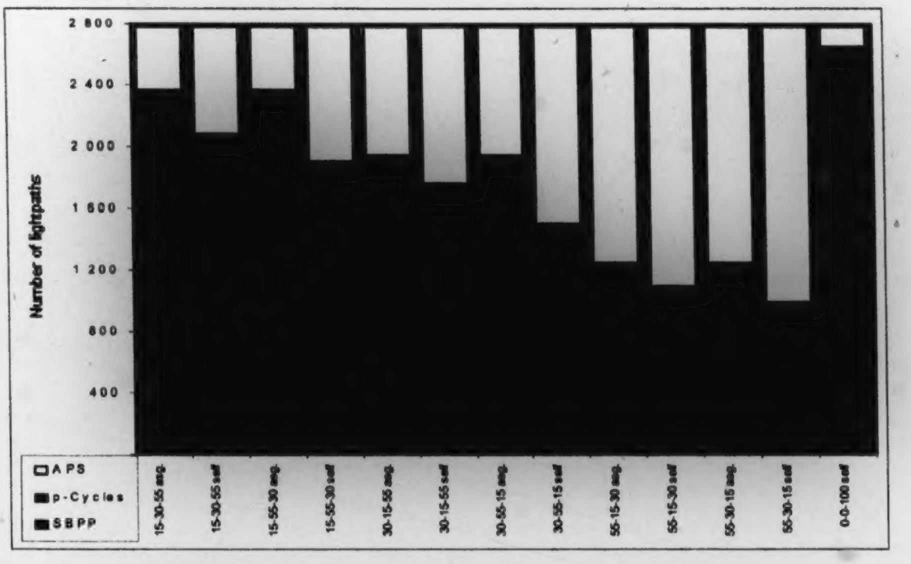

Figure 5.8. **Demand upgrading under self-selecting method design model in Generic U.S. network (from [21]).**

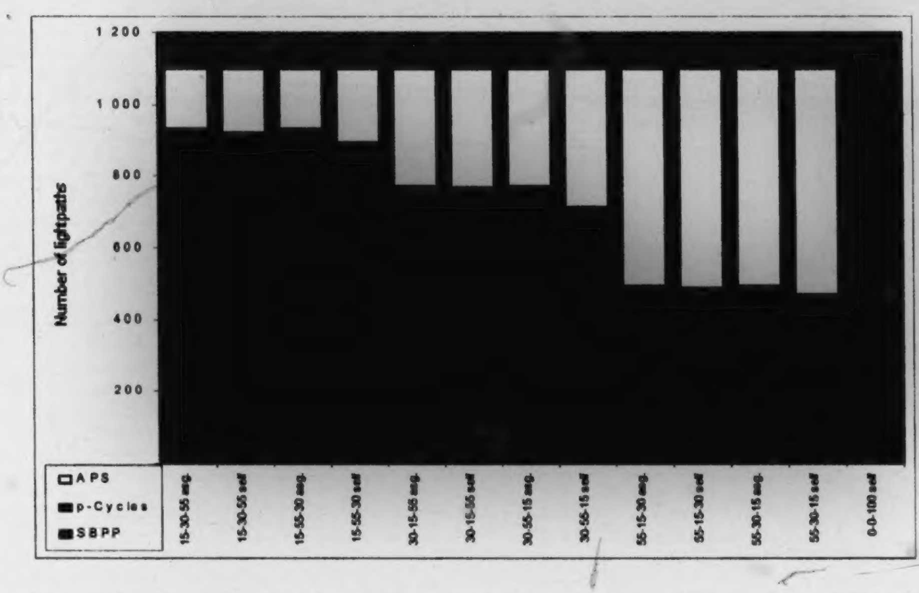

Figure 5.9. **Demand upgrading under self-selecting method design model in COST 239 network (from [21]).**

quite a dramatic shift in some cases. The number of lightpaths protected by *p*-cycles increases in the majority of cases, and additionally, we also see large increases in 1+1

APS protection in the Generic U.S. network. 1+1 APS increases were harder to see in COST 239, although in a few cases there is a slight change there as well.

Why would demands ever move up from an efficient, shared-protection method like *p*-cycles or SBPP to the apparently capacity-wasting 1+1 APS (with at least 100% redundancy)? To answer, we need to focus on *p*-cycles as a shared-protection method. Shared spare capacity leads to efficiencies when many working demands are present. However, imagine the opposite of a "fully-loaded" *p*-cycle – a "barely-loaded" *p*-cycle with only a single working demand. In this case, to protect this single demand, two disjoint paths of spare capacity need to be provisioned (to build the *p*-cycle). In contrast, instead of provisioning a whole cycle to protect this one demand unit, a 1+1 APS arrangement would simply provide a single duplicate path. This difference is illustrated in Figure 5.10. This is how we can justify what the solver appears to be doing – in cases where there are no opportunities to share capacity on the already-built *p*-cycles, and where any potential new *p*-cycles would not be used efficiently, it will be cheaper to just use APS (which the solver is "biased" towards as a preferred performance choice anyway). This is why in COST 239 we saw fewer increases in APS-protected units than in Generic U.S. – because COST 239 is more richly connected, the option of sharing the benefits of a *p*-cycle will be available to more demands. In Generic U.S., however, especially for demands with an East-West orientation, simply using an APS setup on both "sides" of the ladder will sometimes be cheaper than provisioning a collection of *p*-cycles with poorly loaded straddler spans.

While Figure 5.8 and 5.9 indicate the aggregate number of demands that use each method in the before and after scenarios, this is not the same as the number of individual,

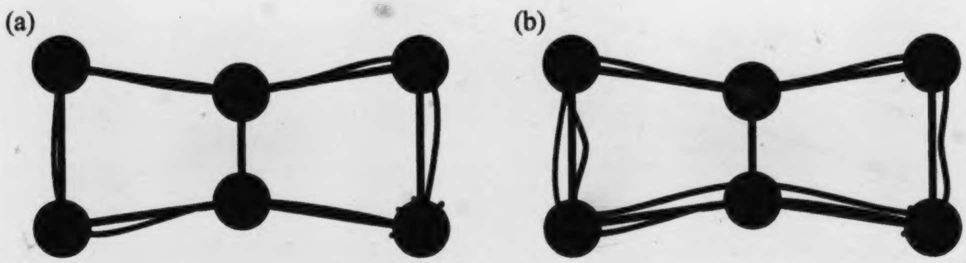

Figure 5.10. For a single incremental working path (in red), a 1+1 APS solution (a) uses less capacity than the solution with a *p*-cycle (b).

specific lightpaths that enjoy a better grade of protection with the self-selected model. For example, if one lightpath moved from SBPP to *p*-cycles, and another moved from *p*-cycles to APS, by looking at the charts we would only see an improvement of one unit in terms of "method number" (APS would have one more unit than before, and all others would be the same or lower.) But in reality, two lightpaths both saw an increase in their quality of protection.

To bring this out more clearly, we list the percentage of individual lightpaths that saw an improvement in their CoP in Table 5.2. These figures show an even more pronounced benefit than the charts because they take into account the individual units moving up. We can see upgrades in approximately 15 to 32% of the time in Generic U.S. and 10 to 24% of the time in COST 239. This is a significant benefit just from applying a new design model, and remarkably, is accomplished even while the overall network capacity cost is being reduced.

In addition to the 15%-30%-55% mixes, for the self-selected model we also tested a 0%-0%-100% split as an interesting "what if?" scenario, on the side. This is a demand pattern of 20 units per O-D pair, all of which *can* be satisfied with the slowest method, SBPP. But the solver, of course, is free to use the best methods possible while minimizing design cost. In the COST 239 case, the bar on the far right side of Figures 5.7 and 5.9 shows however that the minimum requirement is also the cheapest, and all demands are assigned to SBPP. However, in Generic U.S., *a significant portion of lightpaths were served by p-cycles (almost 41%) and a few by APS.* Even though they

Demand mix	Generic U.S.	COST 239
15-30-55	31.9%	24.4%
15-55-30	28.6%	18.0%
30-15-55	29.9%	16.8%
30-55-15	21.9%	11.7%
55-15-30	17.4%	13.4%
55-30-15	15.1%	9.5%
0-0-100	45.4%	0.0%

Table 5.2. Characterization of improved protection available with self-selected method (from [21]).

could have been satisfied with SBPP (which is commonly thought of as a cost-effective, economy-class method), the solver chose to upgrade fully 45.4% to the faster *p*-cycle and APS methods. Figures 5.6 and 5.8 illustrate the split. This self-selected solution actually turns out *lower in cost* than a pure SBPP design. From this we can conclude that the mixture of survivability methods and the use of our integrated design model can sometimes offer better performance in terms of both capacity *and* the survivability "profile" of the network (how many demands can use each method). For some denser networks like COST 239, it may turn out that no benefit exists, but as topologies become sparser, we expect that more and more benefits will get unlocked.

5.5. Generic "Real-World" Test Case

In this section, we used the Generic U.S. network above, but this time with a realistic demand matrix representing what the actual customer demands on the network might look like. The demand split between the three techniques was also based on a realistic assessment. Both the demand quantities and split were provided by Nortel Networks. The exact assignment was 20.5% to APS (60 ms), 23.7% to *p*-cycles (80 ms), and 55.8% to SBPP (200 ms). We can now present the results of a quick check on performance with this new demand pattern and mix. Moving from the assigned case without *p*-cycles to the assigned case with *p*-cycles yielded a modest 2.3% reduction in capacity cost. The reduction from the assigned model with *p*-cycles to the self-selected model with *p*-cycles was 4.8%. As in the previous sections, however, the more significant result is again that a large number of lightpaths were able to upgrade. Here we observed 318 out of 1310 in total (or 24.3%) enjoying a higher grade of protection than in the initially listed set of requirements above.

5.5.1. Effect of Bi-criteria Objective

When our model was first developed, we added the bi-criteria objective term because we thought it could possibly have a positive effect, and that it might be wise to include it just to be sure that the solutions were all biased in the right direction. To quickly review, the first term to minimize in Equation (5.1) is the total capacity cost, while the second term is the weighted sum of all the method numbers used for all demands, with lower numbers representing better methods, as indicated previously in Table 5.1. The weighting factor ε is set to a very low value so that while the second term still has the general effect of "minimizing" the sum of method numbers, it will not be large enough to offset the capacity cost in any significant way.

To test if the bi-criteria term really had any effect and if equivalent cost solutions were rare so that there might not be any point to having it, we solved the Generic U.S. network with the realistic demand pattern above, with and without the second term in the model's objective. The optimality tolerance was specified to the very tight value of 0.001% of optimal so that we could be assured of an accurate result. With the bi-criteria term removed, only 271 lightpaths were upgraded. In contrast, when we included it, a total of 318 demands moved up (as indicated above). The change from 271 to 318 demands took place even with the ε bi-criteria weighting value set to only 0.0001 (the smallest value that CPLEX would treat as a non-zero quantity in our particular setup). Thus, we can see that the bi-criteria term is indeed having an important effect in our model, causing many more demands to enjoy an upgrade in their survivability method than would otherwise be the case.

5.5.2. Introduction of Modularity

We now make a few slight changes to the model to introduce modularity effects (using the methods in [109]) to see what the impact of this will be on self-selected multi-method designs. The changes and additions are as follows:

NEW SET

Y	Set of available module types, indexed by y.

NEW PARAMETERS

c^y	Cost of one module of type y.
τ^y	Size of one module of type y.

NEW VARIABLE

η_j^y	Number of modules of type y placed on span j.

REVISED OBJECTIVE FUNCTION

Minimize: $$\sum_{\forall y \in \boldsymbol{Y}} \sum_{\forall j \in \boldsymbol{S}} c_j \cdot c^y \cdot \eta_j^y + \varepsilon \cdot \sum_{\forall m \in \boldsymbol{M}} \sum_{\forall r \in \boldsymbol{D}} m \cdot \alpha^{m,r} \qquad (5.15)$$

(Minimize *module* cost and use the best methods.)

NEW CONSTRAINT

$$\sum_{\forall y \in \boldsymbol{Y}} \tau^y \cdot \eta_j^y \geq w_j + s_j \qquad \forall j \in \boldsymbol{S} \quad (5.16)$$

(Total modular capacity must support all logical channel requirements on each span.)

We specified the available module sizes as being 1, 4, 8, 16, 32, and 160 channels, and to approximate a "3x2x" economy-of scale effect (3 times the capacity for 2 times the cost – see [109] for details), the costs of these modules were taken as 1, 2, 4, 6, 9, and 25, respectively. With the Generic U.S. network and realistic demand mix, solved to within 0.3% of optimal, we found that the self-selected model with modularity has a higher distance-weighted capacity requirement within the overall envelope of module capacity, but this adjustment is necessary for the solver to minimize the total module cost as much as possible. There was still an improvement, however, over the distance-weighted channel requirements in the assigned model including *p*-cycles. (These distance-weighted capacities we compared did not include any of the unused module overheads.) The total cost of all the modules (as represented in the first objective term) is of course radically different than the objective values we obtained before, because the new value is influenced by the economy-of-scale "discounting" effect (and thus, it is not really "fair" to use this value in any comparisons). As before, however, we noticed the most significant effect in relation to how many demands could upgrade. Where before we saw an improvement of 24.3%, with modularity included we can now observe fully 37.6% of lightpaths enjoying a better CoP.

5.6. Selection of a Single Best Pure Architecture

After developing the self-selected formulation and conducting the tests we just described, we wondered if this new design tool could perhaps be used for a slightly different purpose. The one instance in Figure 5.6 and 5.8 that shows the model determining (on its own) the best mix of protection and restoration methods for the 0%-0%-100% case is especially intriguing. Even though we only indicated that the lowest-performance method was required for each demand (which is essentially no requirement at all, given that all lightpaths need to be protected somehow), the solver found the optimal point on its own. We realized it might be possible to make use of this capability to make decisions *between* method choices in a multi-method framework to determine, scientifically, the best *single* method to protect a given network. (This would be easier than performing separate designs for each method and then manually comparing the results to see which had the lowest cost.) To do this, we would only need to modify our previous model to assert that no more than one method (of the three available) could be used. This is done by adding the following new variable and constraints.

NEW VARIABLE

β^m Equal to 1 if method m is used to protect any amount of demand, 0 otherwise.

NEW CONSTRAINTS

$$\sum_{\forall m \in M} \beta^m = 1 \qquad (5.17)$$

(Only one method can be used.)

$$\Delta \cdot \beta^m \geq \sum_{\forall r \in D} \alpha^{m,r} \qquad \forall m \in M \quad (5.18)$$

(Only the method selected in (5.17) can be used for protection or restoration.)

Because the 0%-0%-100% bar in Figures 5.7 and 5.9 (for COST 239) already showed SBPP being used as a pure architecture when a mix was actually permitted, the COST 239 network was not used with the revised model, because we already know the outcome. We did, however, run it on Generic U.S. and in that case, the result was that SBPP was also selected as the single best architecture ($\beta^3 = 1$).

Because SBPP was the preferred method in both networks, it might be natural to wonder if it will always be the best, or if *p*-cycles might be preferable in some cases. Without exhaustively searching through a number of other topologies to find one where *p*-cycles are chosen, we tested the following "existence-proof" network on a hunch.

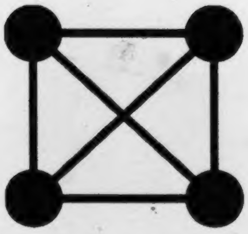

Figure 5.11. "Square" network.

Called "Square" (and depicted in Figure 5.11), this network has 4 nodes and 6 spans, with $\bar{d} = 3.0$. We provided a demand matrix with 1 unit on each side of the square, and 2 units on the two diagonal spans. When used in the *p*-cycle configuration, this would reach the limit of being a fully loaded, semi-homogeneous *p*-cycle. We tested this simple network with the single-architecture model, and *p*-cycles were indeed chosen as the method of protection. While the characterization of the tradeoff between *p*-cycles and SBPP remains an interesting area for future work, we know that it is possible in at least one case for *p*-cycles to be chosen, by the optimal ILP solver, as the theoretically superior method. The self-selected, one-architecture model could be used with other topologies to gain more insight into the conditions when *p*-cycles are the best choice.

5.7. Summary

In this chapter we found that by using a mixed design process, not only can capacity requirements be lowered, but also a significant number of demand units can be upgraded to a better class of protection than they would otherwise enjoy. The bi-criteria objective function ensured that as many demands as possible could do this. We also showed how revising the model can allow it to choose the single best pure architecture. Focusing now more closely on the architecture of primary interest (*p*-cycles), we move on to look at its protection path lengths, design restrictions on path length, and a comparison to the method of imposing path length restrictions by limiting the total cycle circumference.

Chapter 6. Path Length Constraints for *p*-Cycles

As seen previously, a solution strategy for mesh network design is to use Herzberg's idea of limiting the eligible routes provided as an input to the design problem. Not only is this effective in limiting the problem size for the ILP solver, but it may also reflect important real-world limitations on the maximum distance or number of hops in the working or restoration paths. It is also possible, and in many cases necessary (due to a variety of factors), to impose similar limits on the set of eligible *p*-cycles in a design, as we discussed in the literature review in Chapter 3. This restriction on the eligible cycle set can be imposed in several ways. One of the most common is to simply take a certain number of the shortest cycles. Another is to take a certain number of the "best" cycles according to a metric such as *TS* or *AE* (the number of working units associated with a cycle or the number of working units normalized to the number of spare units, respectively). If the motivation is specifically to make the cycle set reflect real-world constraints, all cycles below a stipulated circumference limit could be enumerated as the eligible set. This simple approach, however, does not fully exploit an operational aspect of *p*-cycle protection in that the actual length of each *protection path* is what really matters, and not necessarily the circumference of the cycle from which such paths are derived. In principle, long *p*-cycles could exist in a design while providing short protection paths to straddling spans, and would be admissible to the eligible cycle set under this framework, when otherwise they would be thrown out (thus preventing their consideration by the solver), on the basis of their overall length.

In this chapter we present a model for imposing a direct restriction on the length of *p*-cycle protection paths, which allows us to find optimal designs while taking this factor into account in a very precise way. This allows us to investigate the possible existence of a threshold hop-limit effect for *p*-cycles, and also allows for an accurate comparison of circumference-limited *p*-cycle designs and similarly path- or hop-limited mesh designs. This comparison was previously not possible because the circumference limits applied to *p*-cycles may exclude certain paths of an acceptable length while the hop limits for a span-restorable mesh would not. (The two approaches are not directly equivalent.) We present results to validate the new model, to compare path-limited *p*-cycle design to other techniques, and to obtain new insights about the *p*-cycle method in general. The model, discussion, and results to follow are the result of a collaborative project with Adil Kodian (a colleague and Ph.D. candidate in the TRLabs Network Systems group), and were recently also compiled in the form of a conference paper [110].

6.1. Overview of Path Length Restrictions and Surrounding Issues

In general, optical networks may frequently be designed with some sort of path length restriction in mind. This would usually arise from optical signal quality considerations or to avoid the cost of regenerators, but could also come from network management policies favoring shorter paths for ease of provisioning, maintenance, etc. Shorter paths may also lead to increased availability, and possibly decreased restoration times, depending on the exact restoration mechanism and its signaling methods. Regardless of how these limits come about, it seems reasonable that design tools should have the ability to incorporate controls on maximum protection path lengths. Such restrictions are easily accommodated for span-restorable mesh networks in the now relatively standard methods by Herzberg (as described in [38] and others – see Section 2.3.5 for more details), and their various extensions later developed (the path restoration studies in [31], for example). In the preprocessing step, the eligible working and/or restoration route sets (or cycle sets, for *p*-cycles) can simply be restricted by whatever criteria is desired (hop counts, physical lengths, etc.) before the ILP solver is run. In [106], some previous work examined path length restrictions specifically for mesh networks, but to date this area of inquiry has remained an unanswered question for *p*-cycles.

6.1.1. Circumference vs. Hop Limits for *p*-Cycles

As discussed in earlier chapters, input cycle set restrictions were necessary in some *p*-cycle designs from the very beginning in order to provide the ILP solver with a small enough solution space so that solutions could be obtained in a reasonable time. The foundational design tools (such as those in [40]) simply selected the *x* shortest cycles out of the universe of possibilities. More recent refinements, such as the preselection techniques in [71], endeavored to refine the input set by selecting its members with a cycle efficiency metric (*AE*). In [98], as part of the optimization step, the number of protection relationships that each cycle could have was constrained to try to improve the dual-failure restorability. However, the motivation for all these limiting techniques has been to improve the overall solution quality (i.e. capacity cost), dual-failure restorability, or possibly other factors, but they were *not* a specific attempt to limit the protection path lengths due to optical signal considerations or other factors necessitating short paths. The primary way this has been effected in *p*-cycle networks to date is simply by limiting the eligible cycle set by an overall maximum circumference limit against which all possible cycles are compared. This approach was first used in [59], and in that study, increases in capacity cost were observed as the maximum *p*-cycle circumference was progressively restricted (as might be expected).

This technique does assert a path length limit because obviously no protection path can be longer than any of the cycles admitted for consideration (less the length of the failed span). However, in a *p*-cycle network, the straddling spans are protected with segments of each cycle, which could be considerably shorter than the overall cycle circumference, and therefore acceptable for signal integrity purposes, even if the entire cycle is too long. Thus, we can describe the "circumference-limiting" approach as more of a bounding technique and not a precise and accurate representation of the problem of protection "path-limiting." An exact representation would permit any *p*-cycle to be considered in any protection scenario where the recovery path was adequately short, and correspondingly, would prevent it from being considered for use in a fashion where the recovery path was too long. Without such a technique, we are confounded by an "apples-and-oranges" comparison issue when *p*-cycles are compared to other survivability methods, such as span-restorable mesh (referred to as simply "mesh" throughout this chapter). The point is that a hop limit in a mesh network is not directly comparable to a circumference limit in a *p*-cycle network and thus, in cases where real-world path length issues are thought to be important, no accurate comparison between these two methods is possible. (The one case that remains the exception, where an accurate comparison *is* possible, is when all possible routes and cycles are enumerated in the eligible sets, so that the solver is looking at the entire universe of possibilities. Due to the effect this has on solution times, however, it will not be a practical option in many cases.)

6.1.2. Motivation for Examining Path Length Constraints

We raise the issue of path length constraints first motivated by the theoretical importance of being able to conduct an accurate research comparison. Because span-restorable mesh networks are related to *p*-cycles (which are also a span-protection type of method), there is definitely a relevant comparison to be made, especially in light of prior work that positions the *p*-cycle method in relation to its mesh counterpart (such as [40]). From a practical design standpoint we are not quite sure, *a priori*, whether the extra precision will be important or not in terms of a capacity cost savings, but if this is the case, it will most likely arise in denser networks where there are more opportunities for *p*-cycle path segments to be used for straddling protection when on-cycle protection is infeasible.

However, we also have two other motivations, one of which is simply a pure research question. In the span-restorable mesh context, it was reported in [38] that this type of network exhibits a "threshold hop-limit" effect. This means that when a hop limit is imposed in the design, and progressively relaxed from the most restrictive limit that still permits feasibility all the way to the absence of any hop limit at the opposite end of the scale, that at some point (i.e., the threshold) there will (abruptly) be no further reduction in spare capacity cost (at all) to be achieved by allowing longer eligible routes in the design. This can be exploited as a timesaving measure in subsequent designs, as there is nothing to be gained by running the problem with a hop limit beyond the threshold when we know that no capacity savings can result. Again, because of the similarity to *p*-cycles, it would be interesting and helpful to see if *p*-cycles exhibited a similar threshold hop-limit effect, and how it might compare to the effect seen in mesh networks.

The final motivation is somewhat less concrete, but arises from the interest in the community (previously addressed in Chapter 4) surrounding Hamiltonian *p*-cycles. While Hamiltonians are certainly one framework where we can exploit the benefits of *p*-cycles, their restrictive nature (in terms of one large cycle protecting all spans) may give a mistaken impression that *p*-cycles as a method are not able to consistently provide short paths. In reality, however, even when large *p*-cycles are used to offer protection, design restrictions on path length (that we develop and explore next) will still allow them to be used to protect straddlers with adequately short paths in cases where this may be useful in the context of an overall solution. This study thus has a somewhat demonstrative aspect to it as well in firmly asserting that *p*-cycles are able to provide arbitrarily short restoration paths, by design.

6.1.3. The Potential Efficacy of Path Length Restrictions

Before we go into the details of the study it is helpful to show a quick "existence proof" type of construction to illustrate the benefit we are looking to have our new design model (with path length constraints) exploit or at least quantify. This example (illustrated in Figure 6.1) shows how, in at least one instance, a method with exact constraints imposed on protection path length can outperform (on the basis of capacity requirements) the prior method of circumference limiting. In part (a) we see a simple example network with the working capacity values beside each span (12 units total). The single *p*-cycle in part (b) would be the most efficient design if path length considerations were not important. This solution is quite economical with spare capacity of only eight units. However, once we impose a five-hop limit in part (c), using a circumference limiting approach with the overall length no greater than five, a total of 20 units of spare capacity will be necessary to build the three (now smaller) unique cycles shown. Note that we

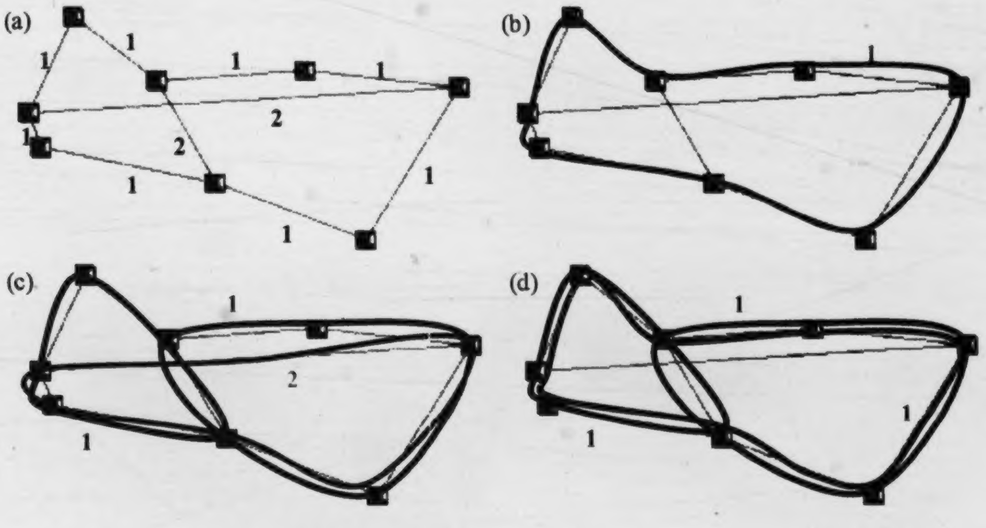

Figure 6.1. Example showing how large *p*-cycles with a path length (hop) limit can be more effective than imposing a simple circumference limit (from [110]).

need two copies of the orange cycle. In contrast, if we keep the length limit of five but replace circumference limiting with explicit path length limiting, we could build the cycle set shown in part (d) as an alternative. One of these cycles is longer than five hops, but is only used to provide protection to the horizontal straddling span, with an acceptable four-hop path on each side. This design has only 18 units of spare. While this is not a huge difference from the 20 units in part (c) of the figure, it does illustrate that, in theory, the use of explicit path length restrictions may be able to unlock a capacity benefit.

6.2. ILP Optimization Model with Path Length Constraints

To find out how often the effect we just illustrated might actually occur in practice, and the extent of any possible savings, we now present our design model that can directly assert a path length constraint. This model could be used in some cases as a replacement for the standard *p*-cycle model with circumference limiting in the preselection process, but in other cases its complexity might make the standard model a better choice, depending on the difference in capacity cost outcomes between the two, which we have yet to quantify. This model is based on a previously published model [71],[79] for joint *p*-cycle design, where the working path routing is done optimally by the ILP solver in conjunction with the *p*-cycle selection decisions.

The mechanism in this model to assert the path length constraints extends prior *p*-cycle models (such as those in [29],[40],[59]) by explicitly considering each "side" of a cycle. Prior models only treated each cycle as a single entity with two protection paths available, but there was no way to tell from the parameters and variables exactly which portion of a cycle was offering protection to a certain span (the $x_{i,p} \in \{0,1,2\}$ values do not actually indicate this). This changes when we consider each side separately. We designate the sides as either left (L) or right (R), and Figure 6.2 illustrates how we label them in both the on-cycle and straddling span failure scenarios. For an on-cycle span, the remaining "side" is simply denoted "R", while in the straddling span case where two sides are available, the longer one is always "R" and the shorter one is "L." (When they are the same length the designation is arbitrary.)

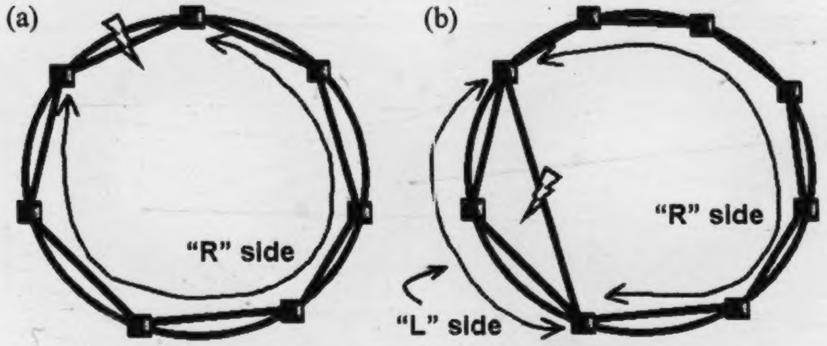

Figure 6.2. Illustration of left (L) and right (R) path definitions used in subsequent ILP model, for (a) on-cycle spans and (b) straddling spans (from [110]).

We split the cycle into its component sides obviously because one side might be an acceptable length while the other might be too long, so by having access in the model to each of them separately we can allow the final design to use any acceptable path while ignoring those that are not. This comes with a price, however, because the number of *p*-cycle variables in the problem is quadrupled as a result. Before attempting to solve this new model we must of course run a preprocessing program to calculate all the input parameters, including enumeration of the eligible cycle set. Our program (again developed using available code when possible, such as [111]) may still need to assert absolute circumference limits as well simply to keep the cycle set to a manageable size. However, we distinguish this from circumference-limiting specifically used as a surrogate for hop-limiting, based on the magnitude of the limitation and the relation (if any) to the hop limit value. Circumference-limiting used as a replacement for hop-limiting will always impose a limit on circumference (C) which is exactly equal to $H + 1$ (one greater than the hop limit). In contrast, any required *absolute* circumference limiting done in the preprocessing step will have no relation to any hop limits that may be specified, and will only have the goal of reducing the solution space to a level the solver can reasonably consider. For example, with $H = 5$, we may need to impose, perhaps, $C = 12$ in advance as an overall input restriction, but this C limit has no relation to the value of H (the hop limit for the design itself), and in addition, is set at a much higher value.

The main extension to the input data set is the addition of new parameters (named $x_i^{p,\mathrm{L}}$ and $x_i^{p,\mathrm{R}}$ in the model) that encode whether or not each protection path offered by each candidate cycle is short enough (or eligible) to protect each possible span failure. For straddling span failures, both parameters could have a 1 or 0 value depending on the circumstances. For on-cycle failures, the R side parameter can likewise reflect details of the situation at hand, but the L side parameter will always be 0, because the L side path for an on-cycle failure is never defined to exist (as in Figure 6.2(a)). For example, if we imposed $H = 6$, for the cycle and failure span in Figure 6.2(a) the parameter values would be $x_i^{p,\mathrm{L}} = 0$ and $x_i^{p,\mathrm{R}} = 1$. The same hop limit applied to the scenario in Figure 6.2(b) would result in both parameters taking a "1" value. But if we set the hop limit to only $H = 4$, now we would see both parameters set to "0" in part (a) (because the on-cycle path is too long), and in part (b), only the shorter left side would be eligible, giving $x_i^{p,\mathrm{L}} = 1$ and $x_i^{p,\mathrm{R}} = 0$.

We can now present our basic model, which will be used as a foundation for the work in the remainder of the chapter:

SETS

$\boldsymbol{Z}$	Set of available module capacities, indexed by *m*.
$\boldsymbol{S}$	Set of spans, indexed by *i* (failed) or *j* (surviving).
$\boldsymbol{D}$	Set of demand relations, indexed by *r*.
$\boldsymbol{P}$	Set of eligible cycles, indexed by *p*.
$\boldsymbol{Q}^r$	Set of eligible working routes for each demand relation *r*, indexed by *q*.

PARAMETERS

Δ	A large positive constant (100000).
c_j^m	Cost of one module of the m^{th} capacity on span *j*.
z^m	Capacity of one module of type *m*.
d^r	Number of demand units for relation *r*.
$\zeta_j^{r,q}$	Equal to 1 if working route *q* (for demand relation *r*) crosses span *j*, 0 otherwise.

$x_i^{p,\mathrm{L}}$	Equal to 1 if the L side of cycle p offers an acceptable protection path for failure of span i, 0 otherwise.
$x_i^{p,\mathrm{R}}$	Equal to 1 if the R side of cycle p offers an acceptable protection path for failure of span i, 0 otherwise.
π_j^p	Equal to 1 if cycle p crosses span j, 0 otherwise.

VARIABLES

η_j^m	Number of capacity modules of type m placed on span j.
w_j	Working capacity placed on span j.
s_j	Spare capacity placed on span j.
$g^{r,q}$	Quantity of demand from relation r that uses route q.
n^p	Number of unit-capacity copies of cycle p in the solution.
n_i^p	Number of copies of cycle p used to protect span i.
$n_i^{p,\mathrm{L}}$	Number of copies of cycle p required for protection of span i, when the L side of the cycle is used.
$n_i^{p,\mathrm{R}}$	Number of copies of cycle p required for protection of span i, when the R side of the cycle is used.

OBJECTIVE FUNCTION

PLJCP: Minimize: $\sum_{\forall m \in Z} \sum_{\forall j \in S} c_j^m \cdot \eta_j^m$ (6.1)

(Minimize total cost of capacity modules placed.)

CONSTRAINTS

$$\sum_{\forall q \in Q^r} g^{r,q} \geq d^r \qquad \forall r \in \boldsymbol{D} \qquad (6.2)$$

(All demands must be routed.)

$$\sum_{\forall r \in D} \sum_{\forall q \in Q^r} \zeta_j^{r,q} \cdot g^{r,q} \leq w_j \qquad \forall j \in \boldsymbol{S} \qquad (6.3)$$

(Place enough working capacity to support the demand.)

$$\sum_{\forall p \in \boldsymbol{P}} \left(x_i^{p,\mathrm{L}} \cdot n_i^{p,\mathrm{L}} + x_i^{p,\mathrm{R}} \cdot n_i^{p,\mathrm{R}} \right) \geq w_i \qquad \forall i \in \boldsymbol{S} \qquad (6.4)$$

(Place enough cycles, considering each side separately, to protect all working units.)

$$\sum_{\forall p \in \boldsymbol{P}} \pi_j^p \cdot n^p \leq s_j \qquad \forall j \in \boldsymbol{S} \qquad (6.5)$$

(Place enough spare capacity to build all *p*-cycles.)

$$\sum_{\forall m \in \boldsymbol{Z}} z^m \cdot \eta_j^m \geq w_j + s_j \qquad \forall j \in \boldsymbol{S} \qquad (6.6)$$

(Place enough modules to support working and spare capacity.)

$$n_i^p \geq n_i^{p,\mathrm{L}} \qquad \forall i \in \boldsymbol{S}, \forall p \in \boldsymbol{P} \qquad (6.7)$$

(Must have more copies than number of paths on the L side, for each failed span.)

$$n_i^p \geq n_i^{p,\mathrm{R}} \qquad \forall i \in \boldsymbol{S}, \forall p \in \boldsymbol{P} \qquad (6.8)$$

(Must have more copies than number of paths on the R side, for each failed span.)

$$n^p \geq n_i^p \qquad \forall i \in \boldsymbol{S}, \forall p \in \boldsymbol{P} \qquad (6.9)$$

(Number of copies of *p*-cycle *p* must be greater than the maximum number required by any one failure.)

$$n_i^{p,\mathrm{L}} \leq \Delta \cdot x_i^{p,\mathrm{L}} \qquad \forall i \in \boldsymbol{S}, \forall p \in \boldsymbol{P} \qquad (6.10)$$

(Use no copies of cycle *p* for protection of span *i* on the L side if the L path is unacceptable, unlimited copies otherwise permitted.)

$$n_i^{p,\mathrm{R}} \leq \Delta \cdot x_i^{p,\mathrm{R}} \qquad \forall i \in \boldsymbol{S}, \forall p \in \boldsymbol{P} \qquad (6.11)$$

(Use no copies of cycle *p* for protection of span *i* on the R side if the R path is unacceptable, unlimited copies otherwise permitted.)

All variables are also non-negative integers. Equations (6.2) and (6.3) ensure that demands are routed and enough working capacity is placed. Equation (6.4) asserts that enough *p*-cycles are selected, and makes use of the new parameters (generated by the preprocessing program in advance, based on a specified hop limit) to consider the protection ability of each side independently. Equation (6.5) calculates spare capacity requirements as in the conventional model, and Equation (6.6) adds modularity (as in [109]). The total module cost is what the objective function is trying to optimize. Equations (6.7)-(6.9) force the total number of *p*-cycles to be greater than or equal to the total number that are required in the event of any possible span failure. And finally, Equations (6.10) and (6.11) are "added valid knowledge" constraints that the problem does not strictly require, but that may help the ILP solver to converge on an acceptable solution faster. These particular constraints immediately force the number of copies of a cycle chosen to protect a certain span with either the L or R side to be zero if the applicable side is too long, as specified by $x_i^{p,\mathrm{L}}$ or $x_i^{p,\mathrm{R}}$.

6.3. Experimental Setup and Results

We now use this model to try to address some of the issues we touched on earlier. We first present our solution method for the comparative mesh reference designs as well as some general parameters and methods.

6.3.1. Mesh-Restorable Comparative Reference Designs

Normally in mesh network design, the hop limit for the eligible working and restoration routes is imposed in the preprocessing program in advance of the ILP process. For repeated varying of the hop limit though, this can be time consuming (to re-generate the data file for every hop-limit value), so for this study we found it to be technically easier to simply modify the standard joint model for mesh design [38], to impose the hop limit directly in the model (as opposed to implementing it in the definition of the sets). The preprocessing program was thus run only once and the single output file it created was given to AMPL as the input data in all cases. To impose the hop limit in the model, we simply added a few new constraints, which are given below.

NEW CONSTRAINTS

$$\gamma_i^u \le \Delta \cdot f_i^u \qquad \forall i \in \boldsymbol{S}, \forall u \in \boldsymbol{U}_i \quad (6.12)$$

$$\gamma_i^u \ge \nabla \cdot f_i^u \qquad \forall i \in \boldsymbol{S}, \forall u \in \boldsymbol{U}_i \quad (6.13)$$

$$\gamma_i^u \in \{0,1\} \qquad \forall i \in \boldsymbol{S}, \forall u \in \boldsymbol{U}_i \quad (6.14)$$

$$\sum_{\forall j \in \boldsymbol{S}, j \ne i} \partial_{i,j}^u \cdot \gamma_i^u \le H \qquad \forall i \in \boldsymbol{S}, \forall u \in \boldsymbol{U}_i \quad (6.15)$$

These constraints include parameters and variables that we have not encountered so far in this thesis. To introduce them, we define U_i as the set of restoration routes eligible to restore span *i*. When span *i* fails, the restoration flow assigned to each route *u* is represented by f_i^u. To specify these restoration routes, the 1/0 parameter $\partial_{i,j}^u$ indicates whether or not each route *u* (for failure of span *i*) crosses surviving span *j*. We add the new binary variable γ_i^u to serve as a 1/0 indication as to whether f_i^u is zero-valued or not (i.e. whether or not a restoration route is used at all, for any amount of flow).

The first two constraints in Equations (6.12) and (6.13) simply map the integer-valued flows in the f_i^u variables into the 1/0 "is there any flow or not" values for the γ_i^us. Equation (6.14) forces γ_i^u to be either 1 or 0. The final constraint, Equation (6.15), is where the path length limit is imposed. This asserts that for every restoration route used ($\gamma_i^u = 1$), the sum of spans on the route ($\partial_{i,j}^u = 1$ for each of them) will be less than the specified hop limit H.

6.3.2. Test Networks and Solution Method Details

Our test networks are shown in Figure 6.3. Most of the results are based on the first four networks ((a)-(d)). The fifth network, in part (e), was reserved for a specific test to be explained later. The first three were synthesized by modifying a common base topology (15n30s1) with arbitrary additions/removals of nodes and spans, while the fourth (in (d)) is the well-known NSFNET study network (from [83]). We used distance-based span costs for the first four networks, and unit costs for the network in part (e) (explained later). (Even if we are designing based on hop-count limits, the actual construction costs for these networks will still be distance-based.) Demand patterns were uniform random in {1-10} lightpaths for each O-D pair.

Other than the specific circumference limits we needed to impose as part of the study, all *p*-cycles were eligible to be included in each solution. In 13n23s there were 501 cycles, 15n26s1 had 871, 12n19s had 127, NSFNET had 139, and 19n35s1 was larger with 10,205 cycles in total. For the input data sets containing eligible working routes (all designs) or restoration routes (mesh reference designs only), we represented all routes

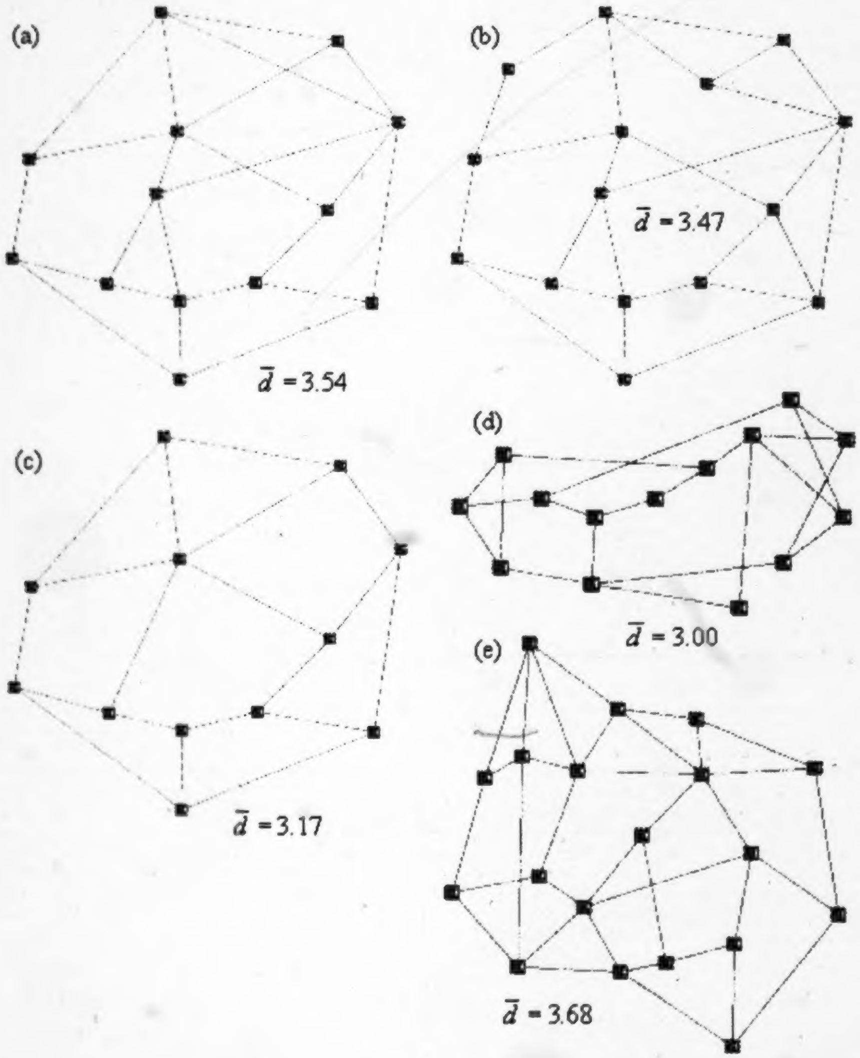

Figure 6.3. Test networks used: (a) 13n23s, (b) 15n26s1, (c) 12n19s, (d) NSFNET, and (e) 19n35s (adapted from [110]).

that were equal in length or shorter than the tenth shortest distinct route (by distance) available. For the mesh reference cases, out of this collection of the shortest eligible restoration routes, the specific hop limit is then imposed in the AMPL model as explained in the previous section. In accordance with our model given above, we solved the main test cases in a modular capacity environment with two module types. These modules had sizes of 12 and 48 channels, with a "4x2x" economy-of-scale effect (see [109] for details). This means that the module of 48 channels, even though four times the size of the first one, will only be twice as expensive. All our test runs were done on "liza" to within at least 1.2% of optimality.

6.3.3. Results: Threshold Hop-Limit Effect

In this section, we made use of the new model above, which implements path length restrictions, as well as the modified mesh network model described in Section 6.3.1. For the circumference-limited designs to follow, we imposed the *C* limit with either a constraint in the AMPL model or with eligible cycle set filtering in preprocessing step, depending on the specific test case (but both techniques impose the same limit).

Let us quickly review the threshold hop-limit effect. As indicated in Section 2.3.5, when the hop limit applied to a span-restorable mesh network is progressively increased (i.e. it becomes more relaxed so that longer and longer paths become available), at some point (which we call the threshold), no further improvement in capacity will be seen even as the hop limit continues to increase. There is thus no advantage to allowing the solver to consider paths longer than the threshold hop limit, as this will not result in any capacity cost improvement and will only lengthen the solution process.

To begin our work in this section, we compared the path-length-limited and standard *p*-cycle designs for the first four test networks, in the case where no limit was imposed. This lets us verify if the new model can achieve the same solution quality as the previous model before we start limiting protection path length. The results were identical in all cases, giving us added confidence in the correctness of our new formulation. We also compared path-limited *p*-cycles with mesh, with no path length limits or route limits asserted (i.e. the full set of cycle candidates and the full set of possible mesh routes). In this scenario, we expect their solutions to be as close as possible. This also turned out to be the case, with solutions being identical (within the range of the 1.2% optimality gap, of course). These results further support the previous claims in many articles suggesting that *p*-cycles are nearly (or just) as good, in the capacity sense, as a span-restorable mesh. We can see the specific data points where *p*-cycles and mesh are almost identical on the far right side of the plot in Figure 6.4. This is where the hop count restriction was unlimited (or "U" as indicated on the figure).

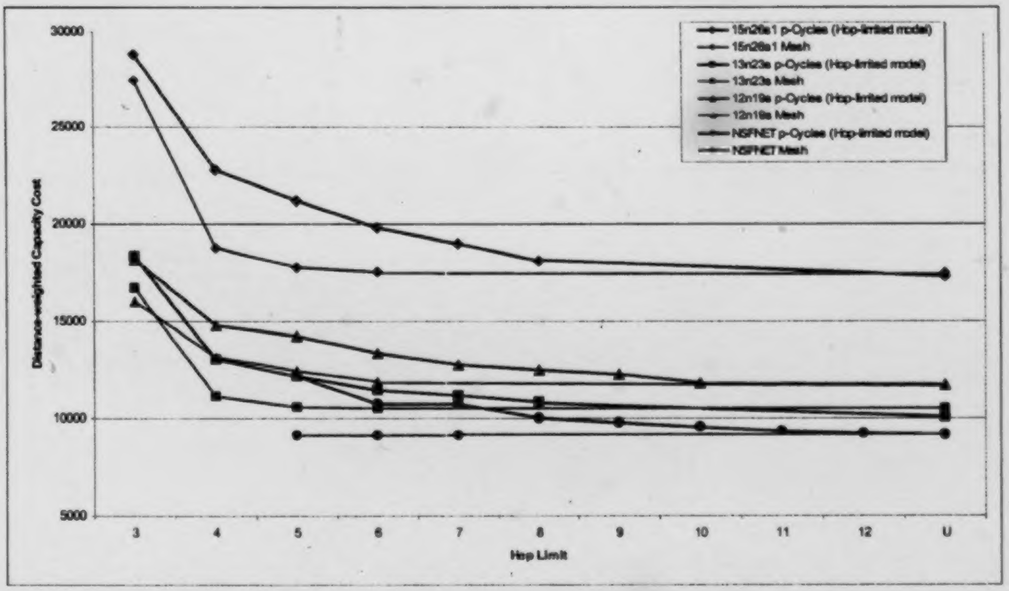

Figure 6.4. Total capacity cost as hop limit increases, showing a threshold hop-limit effect (with designs from both mesh and path-limited *p*-cycle models) (adapted from [110]).

Figure 6.4 also shows what happens as the hop limit is progressively restricted from "U" down to the lowest feasible point for each network. Initially there is not much of a change when moving from right to left along the curves, because we are still above the threshold. However, as we move into the lower hop limits, total capacity cost can increase substantially. For example, for the 15n26s1 network, the threshold hop limit with the mesh appears at six hops. The *p*-cycle threshold for this same network is positioned at roughly the nine-hop point. So from this, we can observe that, first of all, *p*-cycles do seem to exhibit a threshold hop limit, in the same way that mesh networks do. Secondly, we can see that for all four networks in the figure, the *p*-cycle threshold seems to be around three or four hops above the threshold for the mesh. We can also observe that when both designs are above their respective hop-limit thresholds, the *p*-cycle and mesh solutions have essentially similar costs (i.e. overlapping flat lines on the graph). This applies not only in the unlimited cases, but also for *any* hop limit that is above the applicable threshold. An interpretation for this could be that the cost differential between the two is the price for achieving full preconnection with the *p*-cycle structures. In the completely unrestricted ("U") cases, the *p*-cycle designs will always be a subset of the possible mesh designs, because the protection paths arranged in *p*-cycles will always exist as possible mesh restoration paths, but there may be other combinations of mesh paths as well which do not obey the *p*-cycle constraints. It is interesting then to see no cost differential for the unlimited case, even though the full universe of *p*-cycles is already a subset of the universe of mesh routes. But it appears that, in exchange for being able to impose the requirement of preconnected cyclic structures on the universe of possibilities, that the cost for this will become progressively more pronounced as the hop limit becomes more restricted, taking away more and more of the possible ways the ILP solver could try to compensate to keep the differential in costs low.

6.3.4. Results: Comparing to Circumference-Limited Designs

What we look at in this section is the comparison of *p*-cycle path-length-limited (or hop-limited) designs with the corresponding circumference-limited designs (i.e. $C = H + 1$). Circumference limiting is the most common strategy thus far to make sure that no protection path exceeds a maximum value. To see the relative performance, we selected the $H = 6$ point in the 13n23s network used previously. Because an on-cycle span being protected with a six-hop path is excluded from being "counted" toward the six-hop limit (the protection arc will be six hops on the longest *p*-cycle, not the overall circumference), we need to set $C = 7$ in the circumference-limited designs in order to obtain a similar set of cycles. We can see in Figure 6.5 that the distribution of the *p*-cycle paths actually being employed for protection in both cases is very similar. There is a very slight bias toward shorter paths, with results showing a few more two, three, and four hop paths, and slightly fewer paths of five or six hops long. When comparing the solution cost at all hop limits for the four test networks in Figure 6.4, we additionally noticed that all costs were identical (within the 1.2% optimality gap). To summarize, it appears that using path-length-limiting over circumference-limiting leads to neither a capacity improvement, nor a significant shortening of protection paths. It would therefore be a better choice for practical design problems to continue limiting circumference, as the solution times with this approach will be much shorter (due to reduced model complexity), with no real downside in capacity or path length profile.

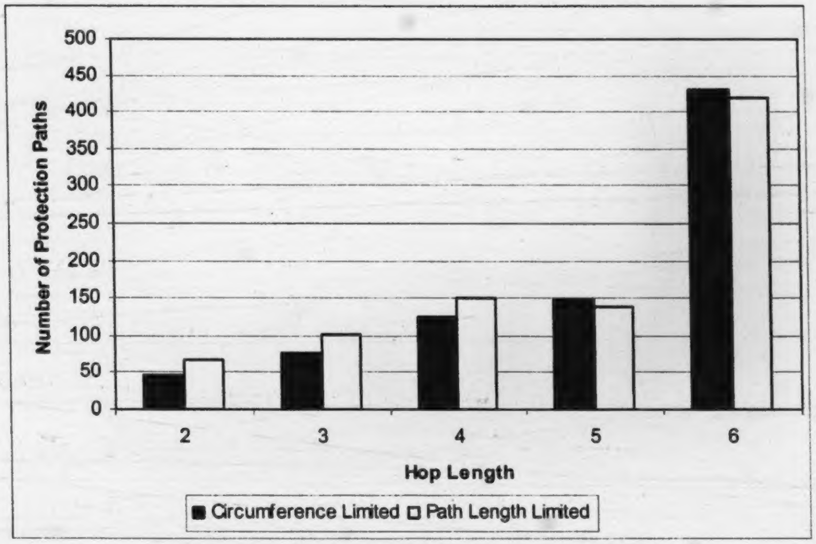

Figure 6.5. Hop-limited path distribution for 13n23s design (*C=7*, *H=6*) (from [110]).

Given this result, we might ask if any working channels (in the straddler configuration) are making use of *p*-cycles that are too long for on-cycle restoration (this is the additional scenario our new model permits over circumference-limited designs). We were able to locate one case for the 15n26s1 network (with a hop limit of six) where an eight-hop *p*-cycle did appear in the solution. Obviously this cycle could not offer protection to any of its on-cycle spans, but was apparently still useful for protection of the straddlers. We thus know that in at least one instance the new model did choose to take advantage of a cycle that would otherwise be "too long."

6.3.5. Results: Checking on a Non-joint, Non-modular Design

Because we used a joint, modular model for all of the prior tests, it occurred to us that perhaps running a follow-up test with a plain, non-modular, spare capacity only (SCP) formulation (with unit span costs) might be wise to check if the aspects of jointness, modularity, or distance-based span costs had any specific effect on the previous results. In this situation we used the fifth network, in Figure 6.3(e), with 19 nodes and 35 spans. Working demands were shortest-path routed in this case because the joint working and spare design method no longer applied.

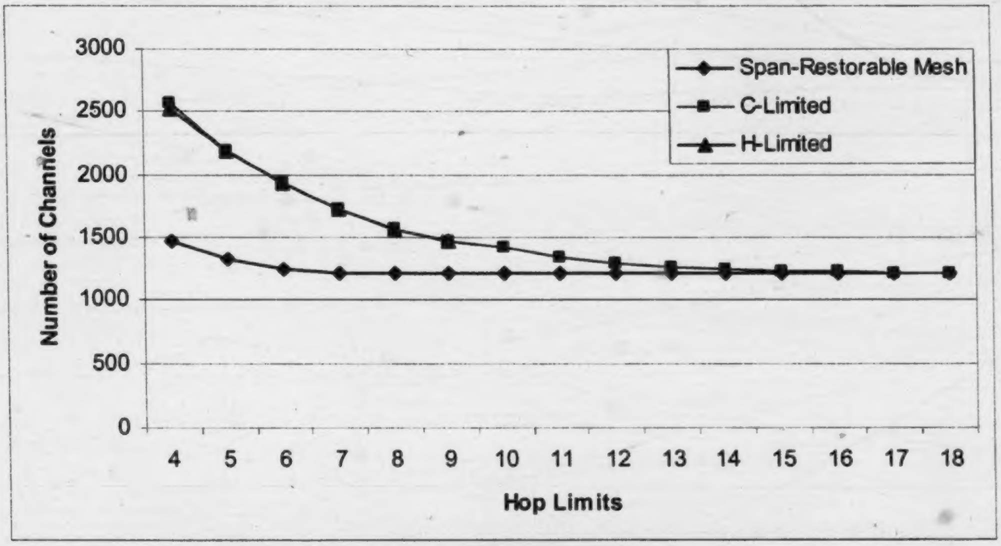

Figure 6.6. **Number of spare channels required as hop limit increases, again showing a threshold hop-limit effect with mesh and both circumference-limited and path-limited *p*-cycle designs (for network 19n35s) (adapted from [110]).**

From Figure 6.6, we can see that the results support the same conclusions reached before, namely, that path-length-limited *p*-cycle designs do exhibit a threshold effect, and that this threshold is not generally reached until several hops after the threshold point for the mesh. For this particular network and test, however, the threshold is reached at about seven hops above the threshold hop-limit of the corresponding span-restorable mesh network. It is also clear here that, again, there is essentially no difference between the hop-limiting and circumference-limiting approaches (the two curves for these are very nearly on top of each other). (Data points on the hop-limited curve were not obtained for the higher *H* values due to computing limitations.)

To check on the path-length distribution in this case, we also collected the data shown in Figure 6.7. This shows the number of six-hop paths in the hop-limited case decreasing, while the shorter three-, four-, and five-hop paths all became more plentiful. This result is similar to the one seen previously in Figure 6.5, and again shows no major advantage to using the more complex hop-limited model when simple circumference-limiting is an effective surrogate. We do recognize, however, that for answering fundamental research questions (as opposed to, say, doing practical network designs), the more complex and rigorous hop-limited model can still be a useful tool.

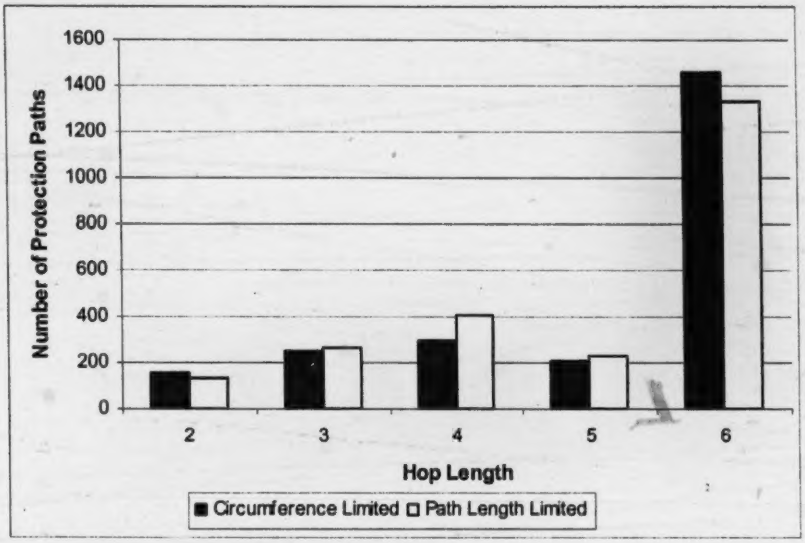

Figure 6.7. Hop-limited path distribution for non-joint, non-modular 19n35s design (*C*=7, *H*=6) (from [110]).

6.3.6. Bi-criteria Technique

In Figures 6.5 and 6.7, we saw the distribution of protection path lengths for the hop-limited and circumference-limited cases. After seeing the rather small improvement that the hop-limiting approach provided, we realized that because it was unlikely the solver would be able to fully "load up" all *p*-cycles, there might be some element of choice in the length of the protection paths chosen. Remembering the bi-criteria approach used successfully in Chapter 5, we decided to use the same idea here as a "what if?" check on how much we could further shift downward the length profile in the hop-limited case.

This was done for the SCP case as in the previous section. The objective function was modified to appear as follows.

REVISED OBJECTIVE FUNCTION

$$\text{Minimize: } \sum_{\forall j\in S} s_j + \varepsilon \cdot \sum_{\forall j\in S} \sum_{\forall p\in P} \left(\phi_i^{p,\mathrm{L}} \cdot n_i^{p,\mathrm{L}} + \phi_i^{p,\mathrm{R}} \cdot n_i^{p,\mathrm{R}} \right) \qquad (6.16)$$

(Minimize spare capacity cost and total hop length of all paths used.)

The $\phi_i^{p,\mathrm{L}}$ and $\phi_i^{p,\mathrm{R}}$ values are simply parameters that specify the length of the protection paths, in hops, on either the L or R side of every *p*-cycle *p* for failure of span *i*.

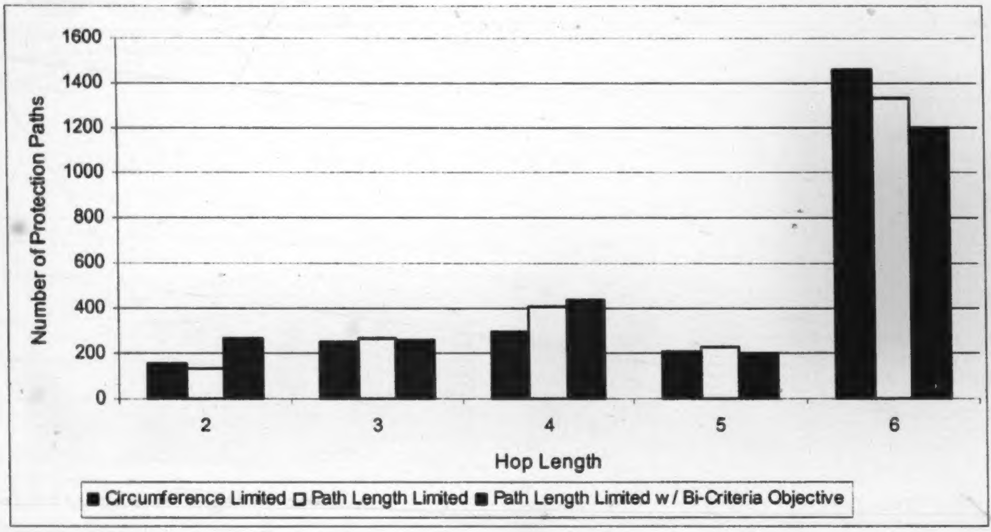

Figure 6.8. Effect of bi-criteria objective in improving path-length-limited results for 19n35s design (*C*=7, *H*=6) (includes previous results from Figure 6.7).

Figure 6.8 illustrates the further improvement available by biasing the hop-limited results towards always using the shortest paths. Like in Chapter 5, we set the value of ε to be 0.0001. For the transition from circumference-limiting to hop-limiting, the average path length value went from 5.08 hops to 4.99 hops. With the bi-criteria "nudge" technique, it further declined to 4.77. At the two ends of the scale, the figure shows a jump in the number of two-hop paths, while the number of six-hop paths declines.

6.4. Summary

In this chapter we looked at path length constraints for *p*-cycles, and found that a threshold effect does exist, similar to mesh networks. We also concluded that in spite of its extra ability to precisely control protection path lengths, the new model may not be a better choice for practical designs, as it offered no significant improvement over the simpler circumference-limiting approach. The complexity of this new model limited our ability to run it on larger networks. This is occasionally the case in other *p*-cycle problems as well, and is sometimes addressed with preoptimization *p*-cycle filtering. In the next chapter, we propose and test a method for *p*-cycle preselection to see how it compares to the other methods currently being used.

Chapter 7. Accurately Distributed Cycle Preselection

As we alluded in Section 3.8.1, the method for determining a good set of candidate *p*-cycles can be extremely important. In our studies in Chapters 4, 5, and 6, we avoided using a preselection method if at all possible and instead opted to give the ILP solver the complete cycle set in order to get accurate and unbiased solutions. This requirement (or preference) for taking the entire cycle set clearly limits our ability to find solutions for larger networks, and makes the solution times for smaller networks unnecessarily long. In this chapter we develop and test a new hypothesis for candidate cycle preselection, which we believe may be superior to other methods known to date.

7.1. Motivation

From the discussion in Section 4.2, we can appreciate that a good candidate cycle set will contain a mixture of both large and small cycles. (This point was also brought up in [88].) In the process of precisely covering a set of working capacities with a layer of *p*-cycles, the large, near-Hamiltonian cycles that tend to originate from the *AE* and especially *TS* metrics are at a disadvantage when coping with high w_i values on a only few individual spans. If only high *AE* or *TS* cycles are present, it will be necessary in some cases to provision a number of these long cycles to protect only a few incremental working capacity units, which could be more efficiently protected instead with some short *p*-cycles in the immediate vicinity. These shorter cycles would act to supplement the larger *p*-cycles that provide very efficient protection to the more evenly distributed "base" quantity of working demand. On the other hand, a candidate set based only on small cycles (such as those originating from the strategy of simply representing the *x* shortest cycles in the network) will likely not work well either. Our discussion of semi-homogenous *p*-cycle networks in Section 4.5 made the point that for *p*-cycles to achieve the theoretical limit on efficiency, a fully loaded Hamiltonian *p*-cycle was required. Although this will usually not be a common solution, as a limiting case it shows that large *p*-cycles that can use the opportunities for straddling span protection are helpful as part of an overall design in order to unlock all the capacity benefits of *p*-cycles.

All of these arguments have been recognized in one form or another in the various strategies for cycle preselection so far proposed. None of them, however, seems to provide a unified approach to incorporate all these competing ideas about what a good candidate set might look like. We can start to do this here by simply stating that it seems a good set must contain both large and small cycles. But should we only represent those cycles that are extremely large or extremely small, or perhaps a group of medium-size cycles too? How many of each should we take, and what should their distribution be?

7.2. The Hypothesis for Statistical Mimicry

7.2.1. Idea of Statistical Mimicry

It seems somewhat natural that we should represent cycles of all sizes, because we know generally that ILP solvers can provide superior performance as the quantity and diversity of possibilities increases. As for their distribution, we could start by considering an unrestricted cycle set where we know the ILP solver could find the perfectly optimal solution with no margin for error coming from the input set. It is generally appreciated in regard to ILP methods (and somewhat intuitive) that normally no single input set element is crucial, in that without it the solution would dramatically increase in cost. Out of thousands of input cycles, if a few are missing, normally the optimization engine will be able to find an equivalent cost solution out of the still large number of combinations that remain. However, within any of the categories of possible cycles (large, medium-size, and small), as we remove more and more elements in that category from the input set, the ability of the ILP solver to compensate with an equivalent arrangement will become progressively more limited, and therefore the solution quality will decrease. (For example, if the number of small cycles declined to the point where they were completely absent from certain regions of the network, this would obviously have a detrimental effect.) Of course, the number of cycles in each category must be limited to some extent (which is the whole point of this chapter, and cycle preselection methods in general), but what we now hypothesize is that the impact will be proportional to the *relative* limitation within each cycle-size category. For example, if there are 1,000 small cycles, and this number is reduced by 900 (or 90%), this may have a much larger impact on the solution compared to removing an equal number of 900 elements from the group of medium-size cycles, which perhaps has, say, 10,000 cycles in total (and thus, the 900 cycles removed would represent a more tolerable 9% of the total). In other words, the key idea is to select a reduced set of candidate cycles that in its own right mimics the overall distribution of cycle sizes in the entire population.

7.2.2. Proposed Preselection Method

Thus, what we propose now is a cycle-limiting method where the statistical distribution of cycle sizes in the limited set is a scaled-down replica of the original distribution in the full set. By reducing the number of elements in each statistical "bin" by the same proportion, the solution space given to the ILP solver will hopefully have the same characteristics as the space when we use a full cycle set. The hypothesis is that we will thus possibly see improved performance with this new method compared to prior strategies. In addition to a comparison with these prior cycle-set limiting techniques, there is also an interesting comparison to be made with the brute-force method of enumerating all cycles. If we maintain the same statistical shape, how close can the performance of the heuristic get to the performance with the full cycle set?

Throughout the current chapter, we use the term "length" to refer specifically to the hop-count, and not the cycle length as determined by the actual span distances. Hop-count length is felt to be a significant factor because it will impact the span coverage of the *p*-cycle set (and thus solution quality) more directly than physical length. Two equal hop-length cycles could have vastly different physical lengths. It is also easier to practically realize in the preprocessing program development. Generally, the lessons we learn from hop-count statistical preselection will be a good foundation for possible work on length-based preselection if this is thought to be a desirable future direction.

7.3. Model and Preselection Program

Because this chapter is only concerned with the preselection strategy, we can use the same basic SCP design model as presented in Section 4.2.1. We also make use of a preprocessing program as before, and for reference purposes employ three programs [100],[111],[112] developed previously by John Doucette and/or William Glenn, both colleagues in the TRLabs Network Systems group. Two of these perform cycle preselection with the *AE* or *TS* metrics (respectively), and take a certain (user-specified) number of the highest scoring cycles according to the metric. The other employs the "vanilla" strategy of taking only the *x* shortest cycles.

The "*x* shortest cycles" program was then extended to come up with a variant to do statistical preselection. It uses the same internal algorithm as in previous versions to now find the entire collection of cycles, up to a maximum of the shortest 10,000,000, a limit approaching the capabilities of the system which we hard-coded as a way of saying "if you can, take all of them." This set of all cycles is then filtered to shape it to the distribution we want before the final set is written to an output file (as in prior versions).

The filtering process begins by counting how many cycles are of each possible circumferential hop-length in the full cycle set. Based on how many cycles in total the user wants in the filtered set (specified as input), the "percentage-to-keep" ratio is calculated and the number of cycles we need in each bin in the new statistically sampled set is calculated as well. Note that all values are rounded up if initially found to be fractional so that the user will get *at least* as many cycles as they asked for, but most likely a few more. This will generally not be significant – for example, in one initial test run, asking for 1,000 cycles actually produced 1,009. This slight supplement will have no practical significance in runtimes, the only issue being that the comparative runs with other preselection methods will need to select exactly 1,009 (for example) for there to be a strictly accurate comparison. The statistical distributions of the full and sampled sets are both written out to information files so that we can capture them for later use.

To populate the new set, cycles are chosen one-by-one, at random, from the full set. If more cycles of that length are needed to satisfy the requirements of statistical proportionality, the individual cycle is added to the new set, the number still required (in that bin) is decremented by one, and the length of that particular cycle in the data structure containing the full set is changed to the artificial value of zero. This will ensure that if the cycle is chosen at random a second time, that it cannot be added to the statistically sampled set, as this new set will never have any need for cycles of length "0." When all bins are full, the new set is sorted and passed to the normal output routine in the main program.

7.4. Test Methods and Results

With this new preselection program in our toolkit, along with previously developed programs and the basic AMPL model, we are now ready to perform our experiments. After the initial checks on a small network just to verify the correct operation of the program, we begin with a test to check the statistical repeatability of our sampling method. To obtain all the results in this chapter, we used our "liza" CPU server as described earlier, but now running CPLEX 9.0 (a newer version than before).

7.4.1. Verification of Statistical Repeatability

Of course, the fundamental issue with using statistical selection methods is that the results could change in detail depending on the exact sequence of random selections. To say that a certain statistically sampled set is better or worse than some other (precisely determined) set is therefore quite tricky because a different statistical sample could result in quite a different answer. Before starting our other experiments it would be interesting to see exactly how susceptible to statistical differences our method is. If the results are all fairly constant at a certain sample size, then we can have more confidence in our later outcomes even though they might only be based on a single run. On the other hand, if they vary quite significantly then we will need to use a suite of data points in all our tests henceforth and represent the performance of the statistical method with an average value and a scatter plot to indicate the full range that was observed.

We chose a 40-node, 60-span network for this test (with $\bar{d} = 3.0$), denoted "40n80s1-60s" and shown in Figure 7.1. A uniform random demand matrix with values between 1 and 10 units per node pair was specified. This network has a total of 71,529 individual cycles, and for our statistical runs, we specified that 2,000 cycles were required. Due to the rounding described earlier when "scaling down" the full set, the preselection program actually returned a sample of 2,022 unique cycles. To see how the variance of the solution values from multiple runs is characterized in relation to other methods, we also found a 2,022-element set based on the *AE* metric, the *TS* metric, and the simple method of choosing the *x* shortest cycles ("Shortest"). These three values, in addition to the solution using the entire 71,529 member set ("Full") and a 50-sample collection from the statistical preselection method ("Stat") give the full collection of results for this test. All solutions were solved to within 0.02% of optimal (or better).

These results are depicted in Figure 7.2, which shows the span-distance-weighted spare capacity cost for each design solution using the standard *p*-cycle model, sorted on the horizontal axis simply by increasing value of cost. Note that the data point for the *TS* method is off the scale that is used (at 1,837,517). This graph shows some unwelcome news – the performance of the statistical preselection approach appears to be highly variable (the largest value is almost 5% more than the smallest). In some cases, it can perform similar to the *AE* approach, but in others it falls behind even the simple Shortest method. What this tells us is that henceforth, we cannot use a single statistical run but instead must use a collection of data points like we did here. (We could also increase the number of cycles in the sampled set, but this moves us away from the goal of finding a method that can offer good performance with only a small sample of the total number of cycles available.) The runtime performance of the statistical sampling method (even for all 50 runs) is also worth noting. A time on CPLEX of 4.72 seconds on average was required for each one, with only 236.2 seconds required for all 50. In contrast, solving the design problem with the full cycle set took 26,382 seconds, or over 10 times longer than all the sampled runs in total.

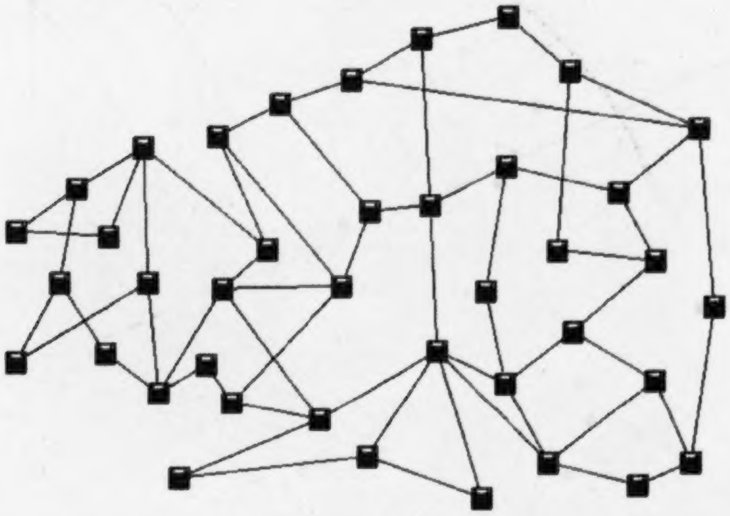

Figure 7.1. Test network used to examine variance effects (40n80s1-60s).

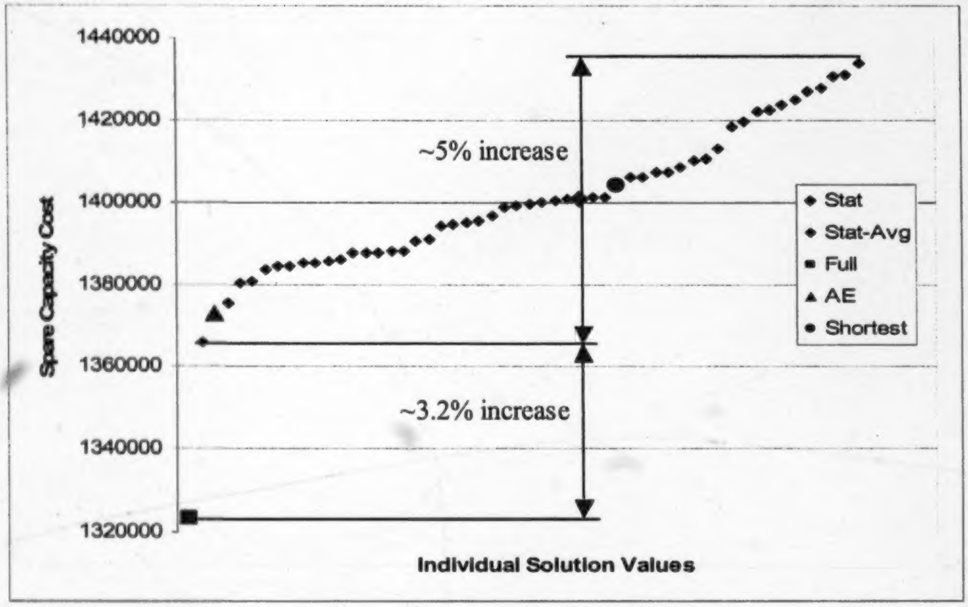

Figure 7.2. Results for variance check with 2,022 cycles in 40n80s1-60s.

7.4.2. Effect of Sample Size

However, we recognize at this point that the number of cycles chosen for this specific test was quite a small percentage of the total number (only 2.83%). It is reasonable to expect that the variance could decrease as the cycle set is increased in size. This next test is therefore simply a repeat of the previous one but with several cycle set sizes to be considered (the data points for the 2,022 cycle case are simply taken from Figure 7.2).

All results in this part were to within 0.03% of optimal (and most to within 0.01%). Figure 7.3 shows how the variance of the statistically sampled results decreases with increasing cycle set size. (We again exclude the *TS* points because they are much higher than the graph scale.) As we reach the 10,019-cycle mark (14% of the total number), it seems a threshold is beginning to emerge where the variance has decreased substantially. We see the values of *AE*, Shortest, and Stat-Avg beginning to converge, and these values are all creeping closer and closer to the theoretical minimum (Full). However, advice from a colleague in the Network Systems group indicated that depending on the exact network, the preselection methods might vary in performance. We thus present the

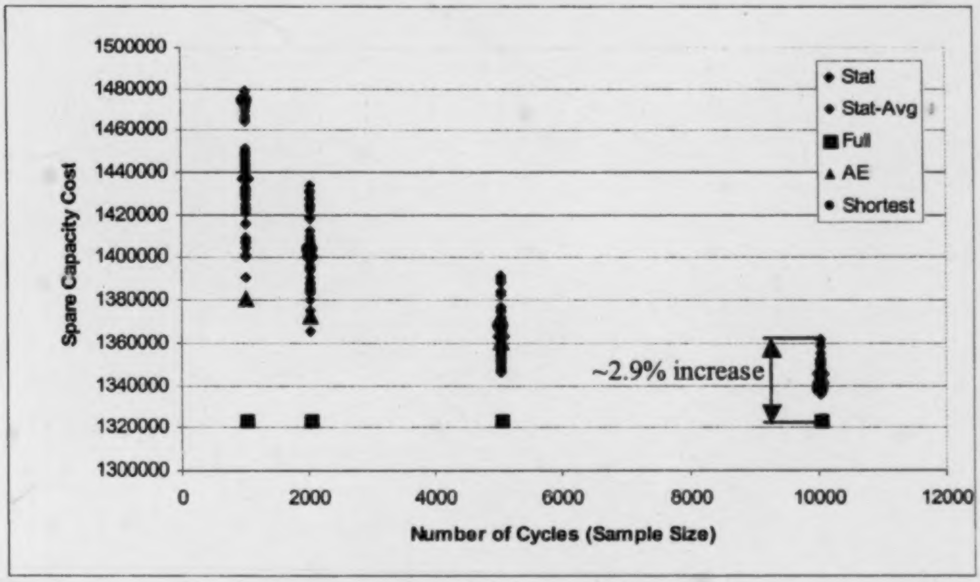

Figure 7.3. Spare capacity results for most methods over varying cycle set sizes (on 40n80s1-60s).

results in Figure 7.3 not as a generically applicable ranking of the different methods, but simply to illustrate the phenomenon of decreasing variance as more cycles are selected, and more generally that variance in the statistically sampled results will continue to be a confounding issue even as the size of the *p*-cycle sets is increased.

7.4.3. Performance Over a Variety of Graphs

The purpose of this next test is to see how the performance varies over a suite of network topologies. For consistency, we set the statistical preselection program to return 15% (or possibly more due to rounding effects) of the total number of cycles available in each network. This will keep the relative effect approximately the same over all networks. From this test, we hope to obtain a more accurate indication of the comparative value of statistical preselection in general.

We chose four test networks, two with a rich topology and two that were sparser. In addition to Generic U.S. and COST 239 from Chapter 5, we also used networks called "15n30s1" and "35n70s1-45s." These are shown in Figure 7.4 below. For COST 239 and Generic U.S., we used their original demand patterns as specified in [58],[103] (converted to the number of lightpaths) and from our colleagues at Nortel Networks, respectively. For the other two, we used a random demand pattern with 1-10 units per node pair.

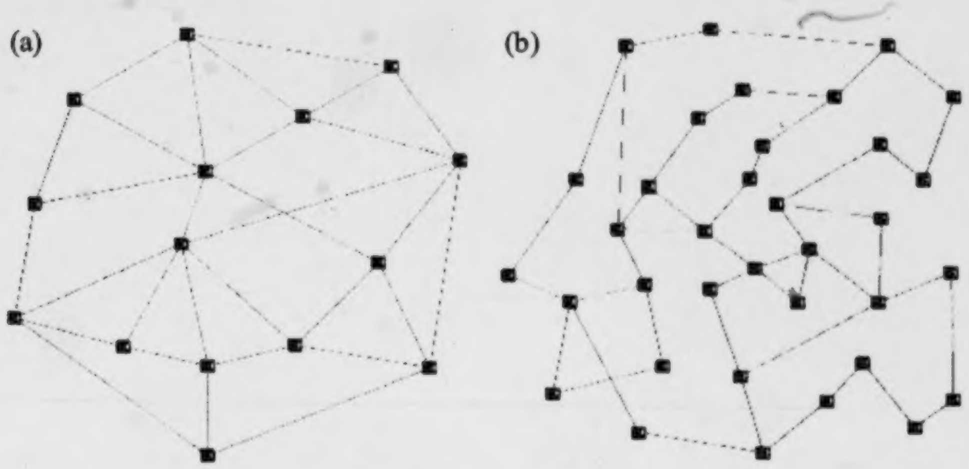

Figure 7.4. Test networks used in addition to those in Chapter 5: (a) 15n30s1, (b) 35n70s1-45s.

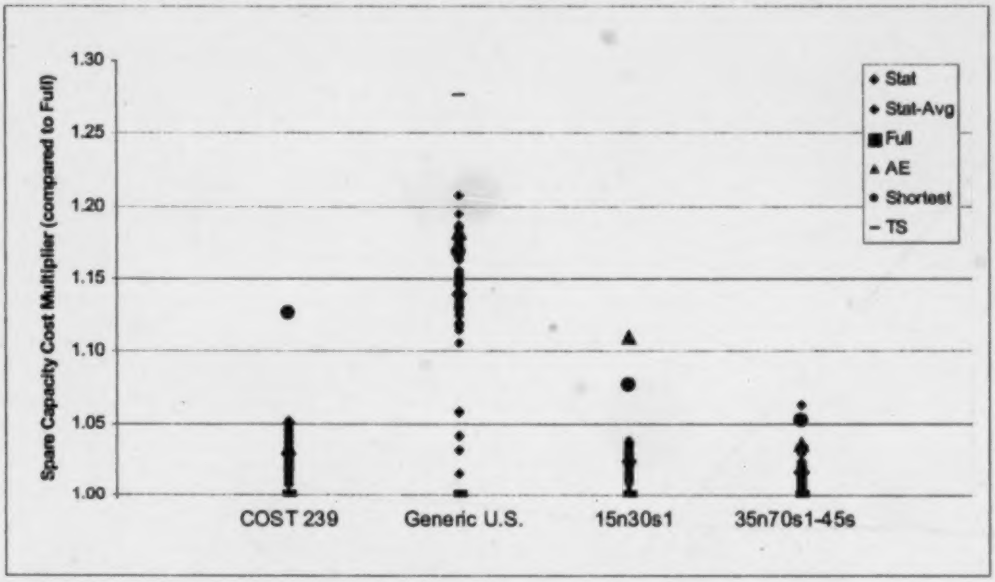

Figure 7.5. Performance (cost premium over optimal) for various network graphs.

The data points in Figure 7.5 show that in general, the new method performs quite well compared to the others, if a number of trials are performed. (The *TS* data point for 15n30s1 could not be obtained because of run-time issues.) The lowest-cost solution discovered in a set of 50 is always very close to the true optimum. However, overall results are still quite variable, especially in the Generic U.S. case. However, we also note that the results for other methods, especially *AE*, are not that consistent sometimes either. It remains up to the network designer in each case to judge whether or not the variance of accurately sampled preselection is appropriate and to either use it or employ other methods, as necessary.

7.4.4. Performance Over Varying Average Nodal Degree

Because of the results above which show good performance in the more connected networks, and slightly poorer performance for Generic U.S. (which is sparser), in this section we used the 20n40s1 suite of network topologies (as in Chapter 4), this time with a varying demand matrix between 1-10 units (randomly chosen) and ran our tests over the range of average nodal degrees in this suite. We only used 10 sample runs in this case instead of 50 to keep the amount of data handled to a manageable level. In each case, the preselection program was instructed to select 15% of all cycles. This became progressively less accurate though for the sparser networks of the family with very small cycle sets (higher relative rounding errors resulted when calculating the size of each bin).

We can see below in Figure 7.6 that the accurately sampled method does better in many cases than the previous methods, although performance is (as before) still variable. At the very upper and lower ends of the scale, the *AE* technique did perform at a comparable or even better level, but for most of the topologies in between, the statistically sampled method did much better. An interesting observation from Figures 7.5 and 7.6 is that while the overall performance of the statistically sampled technique is variable, there is usually one point that is very close to the overall optimum value. Thus, another strategy might be to simply run the statistically sampled process a number of times, and use only the single best result it produces as the final design solution.

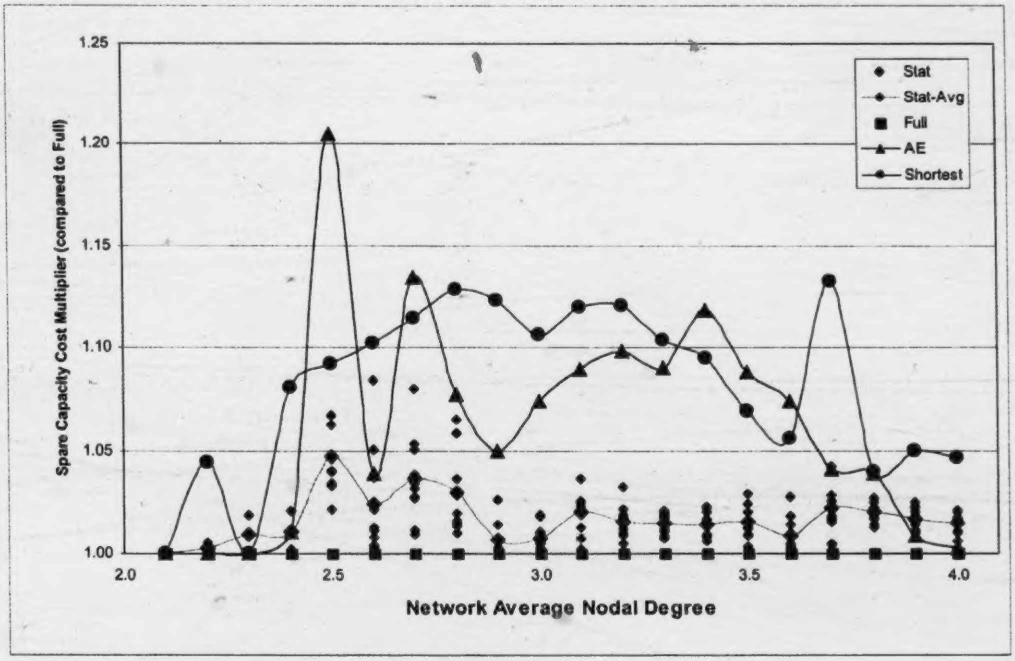

Figure 7.6. Performance over varying nodal degree in 20n40s1 network suite.

7.4.5. Discussion

Our results above are both encouraging on their own and also lead to further ideas about how to obtain an even better preselected set of *p*-cycles. These arise from two main factors that we observed with the statistically sampled method. One of these is the actual distribution of cycles. In Figure 7.7 below, we show the cycle length profile of both the full and statistically sampled sets for the four test networks used in Figure 7.5 above. What we can observe is that although the sampled set is likely taking an adequate number of cycles in the middle of the distribution, the number of small cycles may be suffering. For example, in COST 239, if there are 14 three-hop cycles initially, and this number is reduced to only three cycles after sampling, this restriction may be causing solution quality to suffer, as the very small cycles are the equivalent of the "sand to fit between the rocks." Our first suggestion for refining the method is thus to add, in addition to the statistically sampled set, all the "primary" cycles (as explained in [18],[72]). These primary cycles are the smallest cycles that include each single span

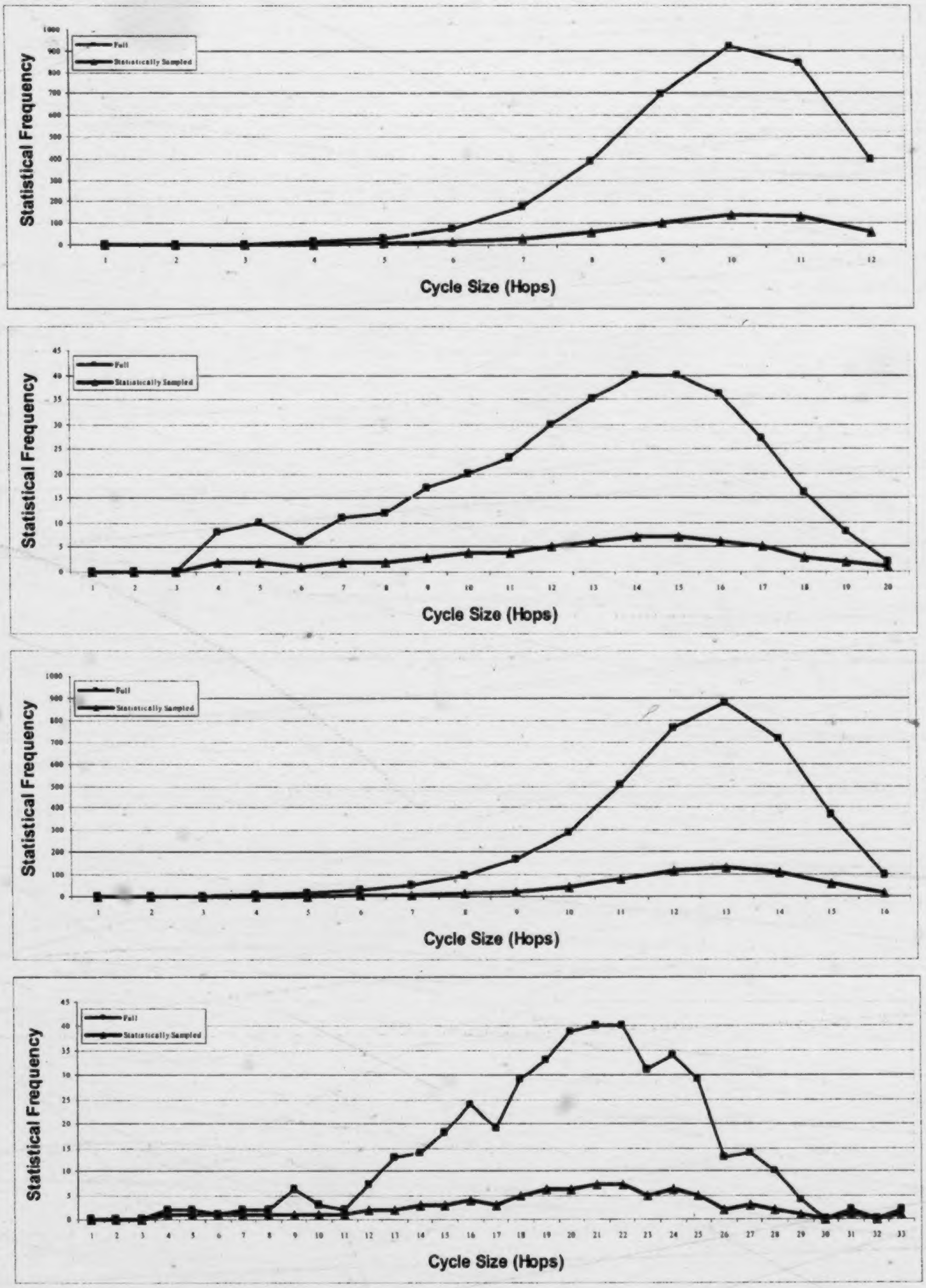

Figure 7.7. Cycle set distributions (top to bottom: COST 239, Generic U.S., 15n30s1, 35n70s1-45s).

as a straddler (although some primary cycles, in order to include the intended span as a straddler, must include another span as well) [18]. For the spans for which a primary cycle does not exist, we could also add the smallest feasible cycle (offering on-cycle protection) in those specific cases as well. The second way we could refine the method is inspired by the variability within the collection of data points. After reflection, we realized that because the method is taking a random sample of cycles at each target bin-size, there is no real mechanism to control the quality of the cycles that are being chosen, nor is there a firmly deterministic algorithm at work to allow us more confidence in the relationship of the final answer to the answers produced by other methods. The suggested refinement is thus to keep the statistical distribution intact, but within the "bin" for each cycle length value, to sort all the cycles of that particular length by the *AE* metric, and then select only the top candidates by *AE* within each bin (however many are required to meet the statistical shape). This would ensure we kept cycles of a variety of lengths, but still had an "elite" collection of them in terms of the number of straddling spans available to be protected.

Another idea in this regard would be to have the accurately-sampled distribution approach not applied to the hop length of each cycle, but rather to the full statistical distribution of *AE* values (*AE*s in the sampled set are a scaled-down sample of *AE*s in the full set). This could be an interesting approach to try as well.

7.5. Summary

In this chapter, we tested and explored a new hypothesis for *p*-cycle preselection. The basic idea is to employ a random sample of cycles that matches the hop-length distribution in the full cycle set. Although the method was found to produce variable results due to the nature of the random sampling, the performance (in terms of solution quality) was quite good in comparison to other preselection methods. This being the last contribution chapter of the thesis, we now move on to the conclusions.

Chapter 8. Conclusions

In this thesis, we explored several interesting ideas within the overarching framework of the *p*-cycle technique for protecting working capacity in an optical transport network. We began by reviewing the basics of transport networking, common protection and restoration techniques, and ILP optimization as applied to network research. Following this we presented five chapters which, as a collection, are intended to serve as both a contribution to our knowledge of the field, as well as a catalyst to further *p*-cycle research in the networking community. The key points from each chapter are reviewed next.

8.1. Main Contributions

In Chapter 3 we offered a comprehensive scholarly review of the *p*-cycle literature available to date. This chapter begins with the initial development of *p*-cycles and proceeds to walk through all the sub-topics and further investigations which were reported in recent years, and should provide the reader with a good overview of the main findings and areas of work in the *p*-cycle field to date.

Chapter 4 began with a demonstration of how Hamiltonian *p*-cycles are not necessarily a better choice when used in a capacitated network graph because a collection of smaller *p*-cycles can more easily arrange themselves to form a strictly optimal solution, a point which is not fully brought out in work that seems to suggest that a Hamiltonian would be a good strategy in all cases [68]. From a more theoretical viewpoint, we showed how a Hamiltonian cycle will not necessarily have the best value of the *AE* metric in certain richly connected networks, and explained the concept of a semi-homogeneous network as a way to fully exploit the protection capacity offered by a *p*-cycle to achieve the lower bound on redundancy for a span-restorable network. In the process of obtaining our illustrative results, we also had an opportunity to present the basic *p*-cycle SCP design model.

Moving on, we explored two main questions in Chapter 5 – first, what effect do *p*-cycles have when introduced to a suite of methods otherwise including 1+1 APS and SBPP, and second, what effect does a "self-selected" design philosophy have on both capacity cost and the number of demands that can upgrade to a method with a faster restoration time. The results showed that by including *p*-cycles in a set of methods, especially one where a large portion of lightpaths can accept a downgrade from APS but are not able to tolerate the SBPP delay, a non-negligible capacity savings can result, making *p*-cycles a viable option for inclusion in the "toolkit" of a carrier. The self-selected model and design approach was found to further reduce capacity requirements while taking advantage of situations where *p*-cycle shared capacity was either warranted but still partially available for protection, or just not warranted at all. In both of these cases, the bi-criteria objective function helped produce results where as many lightpaths as possible were being served by the fastest available technique. We were pleasantly surprised to see that in one test network where we specified that demands could use the lowest CoP, that over 45% of them were nonetheless selected for an upgrade. This is an indication of the value of hybrid methods in unlocking the value present in networks where multiple methods might be used anyway, but with separate designs for each. It also reaffirms the understanding gained from [33] that in general, a truly optimal design will often be a hybrid of several different schemes. For those cases where only one method is desired, however, a modification we proposed to the design model in this chapter allows the optimal method choice and corresponding network design to be determined in a single step by the ILP solver.

In Chapter 6, we dealt with cycle circumference issues as they relate to the prior concept of hop limits in span-restorable mesh networks. After giving an existence proof to illustrate that path length constraints can theoretically be advantageous, we presented a design model that can exactly take individual protection path lengths into account. This model was applied to a small collection of test networks to illustrate the existence of a threshold hop limit with *p*-cycles similar to the same effect seen for a span-restorable mesh. A subsequent test in the chapter also compared path-length-limiting as an approach to the more common but less precise strategy of circumference-limiting, and it was found that in spite of the theoretical differences that there was negligible difference in the results, leaving circumference-limiting still positioned as an effective surrogate for restricting protection lightpath lengths using *p*-cycles. We also applied the bi-criteria "nudge" approach used in Chapter 5 to the distribution of protection paths, and found that it did have an effect in shifting more protection paths to the lower hop-count values. This approach of using a bi-criteria objective to "nudge" solutions without doing a comprehensive "sweep" through all the bi-criteria weighting values may be a useful approach in other situations in networking research as well.

Finally, in Chapter 7, we proposed an approach to preselection in which statistical sampling of the full set of p-cycles was conducted in order to create a smaller set for use in the ILP solution process. This smaller set was different than the cycle sets generated with other preselection methods because it exactly mirrors the distribution of cycle lengths in the full and unrestricted set. We found the performance of the statistical sampling method to be quite variable within the range occupied by some of the other techniques, but overall, our new approach performed quite well on most of the networks tested. We also tested its performance over a suite of networks with varying nodal degree, and found that except for the very highest and lowest $\overline{d}$ values, that the statistically sampled preselection solutions were better than those for all other methods.

8.2. Publications

The following articles were published based on work presented in this thesis:

[21] F. J. Blouin, A. Sack, W. D. Grover, H. Nasrallah, "Benefits of p-cycles in a mixed protection and restoration approach," *Proc. Fourth International Workshop on the Design of Reliable Communication Networks (DRCN 2003)*, Banff, Alberta, Canada, 19-22 Oct. 2003, pp. 203-211.

[70] A. Sack, W. D. Grover, "Hamiltonian p-cycles for fiber-level protection in homogeneous and semi-homogeneous optical networks," *IEEE Network*, vol. 18, no. 2, Mar./Apr. 2004, pp. 49-56.

[110] A. Kodian, A. Sack, W. D. Grover, "p-Cycle network design with hop limits and circumference limits," to appear in *Proc. First International Conference on Broadband Networks (BroadNets 2004)*, San José, California, USA, 25-29 Oct. 2004.

8.3. Further Research Ideas

To further develop the semi-homogeneous network idea outlined in Chapter 4, an interesting study would be one that developed and tested a model which could perform a joint design where the working capacity was routed to "fit" the capacity profile required by a Hamiltonian. The exact Hamiltonian to choose could also be one of the variables in this approach. An interesting general question would be, to what extent can an optimization of this type produce a savings (if possible) over conventional joint *p*-cycle design? Would the advantage of having a fully utilized Hamiltonian cycle for protection be outweighed by the necessary detours on the working routing or would a benefit still arise?

From the mixed design approach in Chapter 5, it would be helpful to quantify the capacity savings and number of demands that can upgrade on either a wider variety of networks or perhaps a network suite as used in Chapters 4 and 7. The other follow-up work that could be undertaken is of course an exploration of the *p*-cycle vs. SBPP tradeoff alluded to at the end of the chapter, to attempt to determine the network properties or other variables that make *p*-cycles a more desirable choice than SBPP in some situations but not in others.

A follow-up experiment to the work on path length constraints in Chapter 6 would be to characterize the threshold hop-limit effect and comparison between circumference- and hop-limiting approaches on larger networks than used in our study. This would further confirm our general findings on a wider range of network sizes. As more computing capability becomes available in the years ahead, this should easily become possible.

And finally, to continue the advances in Chapter 7, the strategy of both sorting cycles within the statistical distribution by *AE* and selecting the top candidates could be implemented and tested, supplemented with the small additional step of adding the shortest cycle to offer protection to each individual span as the extra "sand between the rocks" when large *p*-cycles are used in the majority of the solution. We expect this approach will both outperform the approach we tested here, and also be less variable in the distribution of the ultimate solution results from the ILP solver.

8.4. General Summary

This thesis researched issues in four areas of interest at the time of the author's M.Sc. program:

- Hamiltonian *p*-cycles,
- inclusion of *p*-cycles in a mixed protection and restoration approach with other survivability methods,
- asserting a precise path length limit on *p*-cycle networks, and
- a cycle preselection method with an accurate statistical distribution of candidate cycles.

Some of these investigations were done with colleagues in the TRLabs Network Systems group and in industry, and others were done independently. The topics are connected by the common thread of looking for greater scientific understanding and new technical advances in the area of *p*-cycles. By pushing out the boundaries of what we know about this promising new method, we can both inform our colleagues and stimulate further interest and imagination in advancing *p*-cycles from an idea to an accepted reality in the networking community as a whole.

Bibliography

[1] W. D. Grover, "Network survivability: A crucial issue for the information society," *IEEE Canadian Review*, Summer 1997, pp. 16-21.

[2] Quebec Scientific Information Network, (29 Jan. 2003) [Online], "Forestville-Rimouski underwater cable repaired," Available: http://www.risq.qc.ca/nouvelles/nouvelle_item.php?LANG=EN&ART=1231, Accessed: 20 Jan. 2004.

[3] I. Tham, ZDNet UK, (20 Sep. 2001) [Online], "Anchor-draggers cut Asia's Internet pipe," Available: http://news.zdnet.co.uk/internet/0,39020369,2095715,00.htm, Accessed: 20 Jan. 2004.

[4] G. Wearden, ZDNet UK, (26 Nov. 2003) [Online], "Cable failure hits UK Internet traffic," Available: http://news.zdnet.co.uk/communications/networks/0,39020345,39118125,00.htm, Accessed: 20 Jan. 2004.

[5] L. Bowman, M. Broersma, ZDNet News, (10 Jun. 1998) [Online], "Severed MCI cable cripples the Net," Available: http://zdnet.com.com/2100-11-510740.html, Accessed: 20 Jan. 2004.

[6] W. McAuliffe, ZDNet UK, (3 Aug. 2001) [Online], "Train crash could be to blame for Internet derailment," Available: http://news.zdnet.co.uk/business/0,39020645,2092503,00.htm, Accessed: 20 Jan. 2004.

[7] J. Sosnosky, "Service applications for SONET DCS distributed restoration," *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 1, Jan. 1994, pp. 59-68.

[8] J. C. McDonald, "Public network integrity – Avoiding a crisis in trust," *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 1, Jan. 1994, pp. 5-12.

[9] A. P. Snow, M. W. Thayer, "Defeating telecommunication system fault-tolerant designs," *Proc. Third Information Survivability Workshop (ISW 2000)*, Boston, Massachusetts, USA, 24-26 Oct. 2000.

[10] A. P. Snow, "Network reliability: The concurrent challenges of innovation, competition, and complexity," *IEEE Transactions on Reliability*, vol. 50, no. 1, Mar. 2001, pp. 38-40.

[11] S. Kartalopoulos, "Surviving a disaster," *IEEE Communications Magazine*, vol. 40, no. 7, Jul. 2002, pp. 124-126.

[12] O. Gerstel, R. Ramaswami, "Optical layer survivability: A services perspective," *IEEE Communications Magazine*, vol. 38, no. 3, Mar. 2000, pp. 104-113.

[13] N. Geary, A. Antonopoulos, E. Drakopoulos, J. O'Reilly, J. Mitchell, "A framework for optical network planning under traffic uncertainty," *Proc. Third International Workshop on the Design of Reliable Communication Networks (DRCN 2001)*, Budapest, Hungary, 7-10 Oct. 2001.

[14] W. D. Grover, *Mesh-Based Survivable Networks*. Upper Saddle River, New Jersey, USA: Prentice Hall PTR, 2003.

[15] R. Bhandari, *Survivable Networks: Algorithms for Diverse Routing*. Norwell, Massachusetts, USA: Kluwer, 1999.

[16] W. D. Grover, "The protected working capacity envelope concept: An alternate paradigm for automated service provisioning," *IEEE Communications Magazine*, vol. 42, no. 1, Jan. 2004, pp. 62-69.

[17] M. S. Ryu, H. S. Park, "Survivable network design using restricted *p*-cycle," *Proc. International Conference on Information Networking (ICOIN) 2003*, Jeju Island, Korea, 12-14 Feb. 2003, pp. 918-927.

[18] J. Doucette, D. He, W. D. Grover, O. Yang, "Algorithmic approaches for efficient enumeration of candidate *p*-cycles and capacitated *p*-cycle network design," *Proc.*

Fourth International Workshop on the Design of Reliable Communication Networks (DRCN 2003), Banff, Alberta, Canada, 19-22 Oct. 2003, pp. 212-220.

[19] W. Grover, J. Doucette, M. Clouqueur, D. Leung, D. Stamatelakis, "New options and insights for survivable transport networks," *IEEE Communications Magazine*, vol. 40, no. 1, Jan. 2002, pp. 34-41.

[20] S. De Maesschalck et al., "Pan-European optical transport networks: An availability-based comparison," *Photonic Network Communications*, vol. 5, no. 3, May 2003, pp. 203-225.

[21] F. J. Blouin, A. Sack, W. D. Grover, H. Nasrallah, "Benefits of *p*-cycles in a mixed protection and restoration approach," *Proc. Fourth International Workshop on the Design of Reliable Communication Networks (DRCN 2003)*, Banff, Alberta, Canada, 19-22 Oct. 2003, pp. 203-211.

[22] M. Herzberg, S. J. Bye, "An optimal spare-capacity assignment model for survivable networks with hop limits," *Proc. IEEE Global Telecommunications Conference (GLOBECOM) '94*, San Francisco, USA, 27 Nov.-1 Dec. 1994, pp. 1601-1607.

[23] W. D. Grover, "Self-organizing broadband transport networks," *Proceedings of the IEEE*, vol. 85, no. 10, Oct. 1997, pp. 1582-1611.

[24] W. D. Grover, "Method and apparatus for self-restoring and self-provisioning communication networks," U.S. Patent 4,956,835, 11 Sep. 1990.

[25] D. Zhou, S. Subramanian, "Survivability in optical networks," *IEEE Network*, vol. 14, no. 6, Nov./Dec. 2000, pp. 16-23.

[26] J. Schallenburg, (25-28 Jun. 2001) [Online], "Is 50 ms Restoration Necessary?," Presented at IEEE Bandwidth Management Workshop IX, Available: http://www.ece.ualberta.ca/~grover/pdf/Schallenburg_50ms.ppt, Accessed: 19 Dec. 2003.

[27] F. Blouin, A. Sack, W. D. Grover, H. Nasrallah, "Benefits of *p*-Cycles in a Mixed Protection and Restoration Approach," Draft slides for presentation at Fourth International Workshop on the Design of Reliable Communication Networks (DRCN 2003), Banff, Alberta, Canada, 19-22 Oct. 2003.

[28] W. D. Grover (with D. Morley), "Ring types, sizing, and loading," Draft book chapter for use in E E 681, University of Alberta, Fall 2001.

[29] W. D. Grover, D. Stamatelakis, "Bridging the ring-mesh dichotomy with *p*-cycles," *Proc. Second International Workshop on the Design of Reliable Communication Networks (DRCN 2000)*, Munich, Germany, 9-12 Apr. 2000, pp. 92-104.

[30] D. Morley, "Analysis and design of ring-based transport networks," Ph.D. dissertation, University of Alberta, Edmonton, Alberta, Canada, Spring 2001.

[31] R. R. Iraschko, M. H. MacGregor, W. D. Grover, "Optimal capacity placement for path restoration in STM or ATM mesh-survivable networks," *IEEE/ACM Transactions on Networking*, vol. 6, no. 3, Jun. 1998, pp. 325-336.

[32] W. D. Grover, D. Y. Li, "The forcer concept and its applications to express route planning in mesh survivable networks," *Journal of Networks and Systems Management*, vol. 7, no. 2, Jun. 1999, pp. 199-223.

[33] W. D. Grover, R. G. Martens, "Optimized design of ring-mesh hybrid networks," *Proc. Second International Workshop on the Design of Reliable Communication Networks (DRCN 2000)*, Munich, Germany, 9-12 Apr. 2000, pp. 291-297.

[34] S. Ramamurthy, B. Mukherjee, "Survivable WDM mesh networks, Part I – Protection," *Proc. 18th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '99)*, New York, USA, 21-25 Mar. 1999, vol. 2, pp. 744-751.

[35] J. Doucette, W. D. Grover, "Comparison of mesh protection and restoration schemes and the dependency on graph connectivity," *Proc. Third International*

Workshop on the Design of Reliable Communication Networks (DRCN 2001), Budapest, Hungary, 7-10 Oct. 2001, pp. 121-128.

[36] G. V. Reklaitis, A. Ravindran, K. M. Ragsdell, *Engineering Optimization*. John Wiley & Sons (Wiley-Interscience), 1983.

[37] S. G. Nash, A. Sofer, *Linear and Nonlinear Programming*. McGraw-Hill, 1996.

[38] M. Herzberg, S. J. Bye, A. Utano, "The hop-limit approach for spare-capacity assignment in survivable networks," *IEEE/ACM Transactions on Networking*, vol. 3, no. 6, Dec. 1995, pp. 775-784.

[39] W. D. Grover, D. Stamatelakis, "Self-organizing closed path configuration of restoration capacity in broadband mesh transport networks," *Proc. Second Canadian Conference on Broadband Research (CCBR '98)*, Ottawa, Ontario, Canada, 21-24 Jun. 1998, pp. 145-156.

[40] W. D. Grover, D. Stamatelakis, "Cycle-oriented distributed preconfiguration: Ring-like speed with mesh-like capacity for self-planning network restoration," *Proc. IEEE International Conference on Communications (ICC) '98*, Atlanta, Georgia, USA, 7-11 Jun. 1998, pp. 537-543.

[41] W. D. Grover, "Understanding *p*-cycles, enhanced rings, and oriented cycle covers," *Proc. First International Conference on Optical Communications and Networks (ICOCN 2002)*, Singapore, 11-14 Nov. 2002, pp. 305-308.

[42] W. D. Grover, G. Shen, "Extending the *p*-cycle concept to path-segment protection," *Proc. IEEE International Conference on Communications (ICC) 2003*, Anchorage, Alaska, USA, 11-15 May 2003, pp. 1314-1319.

[43] W. D. Grover, M. H. MacGregor, "Potential for spare capacity preconnection to reduce crossconnection workloads in mesh-restorable networks," *Electronics Letters*, vol. 30, no.3, 3 Feb. 1994, pp. 194-195.

[44] "The role of digital crossconnect systems in transport network survivability," Bellcore, Special Report SR-NWT-002514, Issue 1, Jan. 1993.

[45] W. D. Grover, M. H. MacGregor, "Method for preconfiguring a network to withstand anticipated failures," Canadian Patent Application 2161847, 31 Oct. 1995.

[46] W. D. Grover, M. H. MacGregor, "Method for preconfiguring a network to withstand anticipated failures," U.S. Patent 5,850,505, 15 Dec. 1998.

[47] M. H. MacGregor, W. D. Grover, K. Ryhorchuk, "Optimal spare capacity preconfiguration for faster restoration of mesh networks," *Journal of Network and Systems Management*, vol. 5, no. 2, Jun. 1997, pp. 159-171.

[48] D. Stamatelakis, "Theory and algorithms for preconfiguration of spare capacity in mesh restorable networks," M.Sc. thesis, University of Alberta, Edmonton, Alberta, Canada, 1997.

[49] W. D. Grover, D. Stamatelakis, "Distributed preconfiguration of spare capacity in closed paths for network restoration," Canadian Patent Application 2210207, 11 Jul. 1997.

[50] W. D. Grover, "Distributed preconfiguration of spare capacity in closed paths for network restoration," U.S. Patent 6,421,349, 16 Jul. 2002.

[51] D. Stamatelakis, W. D. Grover, "OPNET simulation of self-organizing restorable SONET mesh transport networks," *Proc. OPNETWORK '98* (CD-ROM), Washington, D.C., USA, 24-25 Apr. 1998, paper 04.

[52] D. Stamatelakis, W. D. Grover, "Rapid span or node restoration in IP networks using virtual protection cycles," *Proc. Third Canadian Conference on Broadband Research (CCBR '99)*, Ottawa, Ontario, Canada, 7-9 Nov. 1999, pp. 33-44.

[53] W. D. Grover, D. Stamatelakis, "Protection of routers in a telecommunications network," Canadian Patent Application 2269649, 22 Apr. 1999.

[54] W. D. Grover, D. Stamatelakis, "Protection of routers in a telecommunications network," U.S. Patent Application, 26 Apr. 1999.

[55] D. Stamatelakis, W. D. Grover, "IP layer restoration and network planning based on virtual protection cycles," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, Oct. 2000, pp. 1938-1949.

[56] D. Stamatelakis, W. D. Grover, "Theoretical underpinnings for the efficiency of restorable networks using preconfigured cycles ('*p*-cycles')," *IEEE Transactions on Communications*, vol. 48, no. 8, Aug. 2000, pp. 1262-1265.

[57] D. Stamatelakis, W. D. Grover, "Network restorability design using pre-configured trees, cycles, and mixtures of pattern types," TR*Labs*, Edmonton, Alberta, Canada, Technical Report TR-1999-05, 30 Oct. 2000.

[58] C. G. Gruber, "Deployment of *p*-cycles in WDM networks," Dipl.-Ing. thesis, Technical University of Munich, Munich, Germany, Aug. 2001.

[59] D. A. Schupke, C. G. Gruber, A. Autenrieth, "Optimal configuration of *p*-cycles in WDM networks," *Proc. IEEE International Conference on Communications (ICC) 2002*, New York City, USA, 28 Apr.-2 May 2002, pp. 2761-2765.

[60] C. G. Gruber, D. A. Schupke, "Capacity-efficient planning of resilient networks with *p*-cycles," *Proc. Tenth International Telecommunication Network Strategy and Planning Symposium (Networks 2002)*, Munich, Germany, 23-27 Jun. 2002.

[61] D. A. Schupke, "Fast and efficient WDM network protection using *p*-cycles," *Proc. IEEE LEOS Summer Topicals 2002*, Mont Tremblant, Quebec, Canada, 15-17 Jul. 2002, pp. 47-48.

[62] C. Mauz, "p-cycle Protection in wavelength routed networks," *Proc. Seventh IFIP Working Conference on Optical Network Design and Modelling (ONDM 2003)*, Budapest, Hungary, 3-5 Feb. 2003.

[63] M. Scheffel, "Configuration of *p*-cycles in optical networks with partial wavelength conversion," Dipl.-Ing. thesis, Technical University of Munich, Munich, Germany, Aug. 2002.

[64] D. A. Schupke, M. C. Scheffel, W. D. Grover, "An efficient strategy for wavelength conversion in WDM *p*-cycle networks," *Proc. Fourth International Workshop on the Design of Reliable Communication Networks (DRCN 2003)*, Banff, Alberta, Canada, 19-22 Oct. 2003, pp. 221-227.

[65] D. A. Schupke, M. C. Scheffel, W. D. Grover, "Configuration of *p*-cycles in WDM networks with partial wavelength conversion," *Photonic Network Communications*, vol. 6, no. 3, Nov. 2003, pp. 239-252.

[66] H. Huang, J. A. Copeland, "Hamiltonian cycle protection: A novel approach to mesh WDM optical network protection," *Proc. IEEE Workshop on High Performance Switching and Routing (HPSR) 2001*, Dallas, Texas, USA, 29-31 May 2001, pp. 31-35.

[67] H. Huang, J. A. Copeland, "Multi-domain mesh optical network protection using Hamiltonian cycles," *Proc. IEEE Workshop on High Performance Switching and Routing (HPSR) 2002*, Kobe, Hyogo, Japan, 26-29 May 2002, pp. 83-87.

[68] H. Huang, J. A. Copeland, "A series of Hamiltonian cycle-based solutions to provide simple and scalable mesh optical network resilience," *IEEE Communications Magazine*, vol. 40, no. 11, Nov. 2002, pp. 46-51.

[69] M. S. Ryu, H. S. Park, "RPC (Restricted p-cycle by Hamiltonian) considering QoP," *Proc. Conference on the Optical Internet (COIN) 2003 and 28th Australian Conference on Optical Fibre Technology (ACOFT 2003)*, Melbourne, Australia, 13-16 Jul. 2003, paper 181.

[70] A. Sack, W. D. Grover, "Hamiltonian *p*-cycles for fiber-level protection in homogeneous and semi-homogeneous optical networks," *IEEE Network*, vol. 18, no. 2, Mar./Apr. 2004, pp. 49-56.

[71] W. D. Grover, J. Doucette, "Advances in optical network design with *p*-cycles: Joint optimization and pre-selection of candidate *p*-cycles," *Proc. IEEE LEOS Summer Topicals 2002*, Mont Tremblant, Quebec, Canada, 15-17 Jul. 2002, pp. 49-50.

[72] H. Zhang, O. Yang, "Finding protection cycles in DWDM networks," *Proc. IEEE International Conference on Communications (ICC) 2002*, New York City, USA, 28 Apr.-2 May 2002, pp. 2756-2760.

[73] D. Stamatelakis, W. D. Grover, "Scalable network restoration device," Canadian Patent Application 2280981, 27 Aug. 1999.

[74] D. Stamatelakis, W. D. Grover, "Scalable network restoration device," U.S. Patent 6,404,734, 11 Jun. 2002.

[75] M. Clouqueur, W. D. Grover, D. Leung, O. Shai, "Mining the rings: Strategies for ring-to-mesh evolution," *Proc. Third International Workshop on the Design of Reliable Communication Networks (DRCN 2001)*, Budapest, Hungary, 7-10 Oct. 2001, pp. 113-120.

[76] W. D. Grover, M. A. H. Clouqueur, K. K. Leung, "Evolution of a telecommunications network from ring to mesh structure," Canadian Patent Application 2392123, 28 Jun. 2002.

[77] W. D. Grover, M. A. H. Clouqueur, K. K. Leung, "Evolution of a telecommunications network from ring to mesh structure," U.S. Patent Application 20030016623, 28 Jun. 2002.

[78] W. Grover, "*p*-Cycles, ring-mesh hybrids and 'ring-mining:' Options for new and evolving optical networks," *Proc. Optical Fiber Communication Conference (OFC) 2003*, Atlanta, Georgia, USA, 24-27 Mar. 2003, pp. 201-203.

[79] A. Kodian, W. D. Grover, J. Slevinsky, D. Moore, "Ring-mining to *p*-cycles as a target architecture: Riding demand growth into network efficiency," *Proc. 19th*

National Fiber Optic Engineers Conference (NFOEC 2003), Orlando, Florida, USA, 7-11 Sep. 2003, pp. 1543-1552.

[80] G. Shen, W. D. Grover, "Extending the *p*-cycle concept to path segment protection for span and node failure recovery," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 8, Oct. 2003, pp. 1306-1319.

[81] D. Rajan, A. Atamtürk, "Survivable network design: Routing of flows and slacks," in *Telecommunications Network Design and Management*, G. Anandalingam, S. Raghavan, Eds., pp. 65-81, Boston, Massachusetts, USA: Kluwer Academic Publishers, 2002.

[82] C. G. Gruber, "Resilient networks with non-simple *p*-cycles," *Proc. Tenth International Conference on Telecommunications (ICT 2003)*, Papeete, Tahiti, French Polynesia, 23-28 Feb. 2003, pp. 1027-1032.

[83] G. Shen, W. D. Grover, "Exploiting forcer structure to serve uncertain demands and minimize redundancy of *p*-cycle networks," *Proc. Fourth International Conference on Optical Networking and Communications (OptiComm 2003)*, Dallas, Texas, USA, 13-17 Oct. 2003.

[84] M. Clouqueur, W. D. Grover, "Mesh-restorable networks with complete dual failure restorability and with selectively enhanced dual-failure restorability properties," *Proc. Third International Conference on Optical Networking and Communications (OptiComm 2002)*, Boston, Massachusetts, USA, 29 Jul.-2 Aug. 2002, pp. 1-12.

[85] D. A. Schupke, "The tradeoff between the number of deployed *p*-cycles and the survivability to dual fiber duct failures," *Proc. IEEE International Conference on Communications (ICC) 2003*, Anchorage, Alaska, USA, 11-15 May 2003, pp. 1428-1432.

[86] D. A. Schupke, "Multiple failure survivability in WDM networks with *p*-cycles," *Proc. IEEE International Symposium on Circuits and Systems (ISCAS) 2003*, Bangkok, Thailand, 25-28 May 2003, pp. 866-869.

[87] M. Clouqueur, "Availability of service in mesh-restorable transport networks," Ph.D. Dissertation, University of Alberta, Edmonton, Alberta, Canada, Spring 2004.

[88] J. Kang, M. J. Reed, "Bandwidth protection in MPLS networks using *p*-cycle structure," *Proc. Fourth International Workshop on the Design of Reliable Communication Networks (DRCN 2003)*, Banff, Alberta, Canada, 19-22 Oct. 2003, pp. 356-362.

[89] L. Lipes, "Understanding the trade-offs associated with sharing protection," *Proc. Optical Fiber Communication Conference (OFC) 2002*, Anaheim, California, USA, 17-22 Mar. 2002, pp. 786-787.

[90] M. Jaeger, R. Huelsermann, D. A. Schupke, R. Sedlak, "Evaluation of novel resilience schemes in dynamic optical transport networks," *Proc. Asia-Pacific Optical and Wireless Communications (APOC) 2002*, Shanghai, China, 13-18 Oct. 2002, pp. 82-93.

[91] D. A. Schupke, M. Jäger, R. Hülsermann, "Comparison of resilience mechanisms for dynamic services in intelligent optical networks," *Proc. Fourth International Workshop on the Design of Reliable Communication Networks (DRCN 2003)*, Banff, Alberta, Canada, 19-22 Oct. 2003, pp. 106-113.

[92] G. Birkan, J. Kennington, E. Olinick, A. Ortynski, G. Spiride, "Making a case for using integer programming to design DWDM networks," *Optical Networks Magazine*, vol. 4, no. 6, Nov./Dec. 2003, pp. 107-120.

[93] J. A. Fee, "Method and system for optical restoration tributary switching in a fiber network," U. S. Patent 5,884,017, 16 Mar. 1999.

[94] G. Ellinas, A. G. Hailemariam, T. E. Stern, "Protection cycles in mesh WDM networks," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, Oct. 2000, pp. 1924-1937.

[95] M. Médard, R. A. Berry, S. G. Finn, W. He, S. S. Lumetta, "Generalized loop-back recovery in optical mesh networks," *IEEE/ACM Transactions on Networking*, vol. 10, no. 1, Feb. 2002, pp. 153-164.

[96] A. Alfa, G.-Q. Wang, "Cycle-based restoration algorithm with bandwidth and flow considerations for mesh networks: developing the objective function," Presented at IEEE Global Telecommunications Conference (GLOBECOM) 2003, San Francisco, USA, 1-5 Dec. 2003.

[97] D. A. Schupke, "An ILP for optimal *p*-cycle selection without cycle enumeration," *Proc. Eighth IFIP Working Conference on Optical Network Design and Modelling (ONDM 2004)*, Ghent, Belgium, 2-4 Feb. 2004.

[98] D. A. Schupke, W. D. Grover, M. Clouqueur, "Strategies for enhanced dual failure restorability with static or reconfigurable *p*-cycle networks," *Proc. IEEE International Conference on Communications (ICC) 2004*, Paris, France, 20-24 Jun. 2004, pp. 1628-1633.

[99] W.-D. Zhong, Z.-R. Zhang, "Design of survivable WDM networks with shared-p-cycles," *Proc. Optical Fiber Communication Conference (OFC) 2004*, Los Angeles, California, USA, 22-27 Feb. 2004, pp. TuP4-1-TuP4-3.

[100] J. Doucette, "PCycle-SCP-DatPrep version 1.0," TRLabs Proprietary Software, Dec. 2001.

[101] H. Liu, O. Yang, S. Shah-Heydari, "A tree-based link protection algorithm," *Proc. Canadian Conference on Electrical and Computer Engineering (CCECE) 2003*, Montreal, Quebec, Canada, 4-7 May 2003, pp. 939-942.

[102] W. D. Grover, J. Doucette, "Design of a meta-mesh of chain subnetworks: Enhancing the attractiveness of mesh-restorable WDM networking on low

connectivity graphs," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 1, Jan. 2002, pp. 47-61.

[103] P. Batchelor et al., "Ultra high capacity optical transmission networks: Final report of action COST 239," Faculty of Electrical Engineering and Computing, University of Zagreb, 1999.

[104] P. Demeester et al., "Resilience in multilayer networks," *IEEE Communications Magazine*, vol. 37, no. 8, Aug. 1999, pp. 70-76.

[105] F. Mobiot, B. Sansò, A. Girard, "A method for the integrated design of reliable GMPLS networks," *Proc. Fourth International Workshop on the Design of Reliable Communication Networks (DRCN 2003)*, Banff, Alberta, Canada, 19-22 Oct. 2003, pp. 53-60.

[106] J. Doucette, W. D. Grover, T. Bach, "Bi-criteria studies of mesh network restoration path-length versus capacity tradeoffs," *Proc. Optical Fiber Communication Conference (OFC) 2001*, Anaheim, California, USA, 17-22 Mar. 2001, pp. TuG2-1-TuG2-3.

[107] W. D. Grover, M. Clouqueur, "Span-restorable mesh network design to support multiple quality of protection (QoP) service-classes," *Proc. First International Conference on Optical Communications and Networks (ICOCN 2002)*, Singapore, 11-14 Nov. 2002, pp. 321-323.

[108] D. A. Dunn, W. D. Grover, M. H. MacGregor, "A comparison of *k*-shortest paths and maximum flow methods for network facility restoration," *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 1, Jan. 1994, pp. 88-99.

[109] J. Doucette, W. D. Grover, "Influence of modularity and economy-of-scale effects on design of mesh-restorable DWDM networks," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, Oct. 2000, pp. 1912-1923.

[110] A. Kodian, A. Sack, W. D. Grover, "*p*-Cycle network design with hop limits and circumference limits," to appear in *Proc. First International Conference on*

Broadband Networks (BroadNets 2004), San José, California, USA, 25-29 Oct. 2004.

[111] J. Doucette, W. Glenn, "PCycleMetric-SCP-DatPrep version 2.0," TRLabs Proprietary Software, Jun. 2002.

[112] J. Doucette, W. Glenn, "PCycle-SCP-AbsScore-DatPrep version 1.0," TRLabs Proprietary Software, Jan. 2002.

Appendix A. Test Networks

A.1. Canada Network

A.1.1. Topology (from [101])

NODE	X	Y	SIZE
0	79	34	4
1	205	97	4
2	104	133	4
3	10	238	3
4	72	191	3
5	172	238	4
6	261	199	6
7	313	259	3
8	219	296	3
9	379	243	2
10	412	179	4
11	447	93	2
12	288	54	4

SPAN	O	D	LENGTH	UNITCOST
0-1	0	1	140.872	140.872
0-2	0	2	102.108	102.108
0-3	0	3	215.353	215.353
0-12	0	12	209.955	209.955
1-2	1	2	107.224	107.224
1-6	1	6	116.362	116.362
1-12	1	12	93.477	93.477
2-4	2	4	66.242	66.242
2-6	2	6	170.309	170.309
3-4	3	4	77.801	77.801
3-5	3	5	162	162
4-5	4	5	110.494	110.494
5-6	5	6	97.17	97.17
5-8	5	8	74.653	74.653
6-7	6	7	79.398	79.398
6-8	6	8	105.702	105.702
6-10	6	10	152.319	152.319
7-8	7	8	101.02	101.02
7-9	7	9	67.912	67.912
9-10	9	10	72.007	72.007
10-11	10	11	92.849	92.849
10-12	10	12	176.071	176.071
11-12	11	12	163.713	163.713

A.1.2. Demand Matrix (Unit Demand Between Node Pairs)

O	D	NBUNITS
0	1	1
0	2	1
0	3	1
0	4	1
0	5	1
0	6	1
0	7	1
0	8	1
0	9	1
0	10	1
0	11	1
0	12	1
1	2	1
1	3	1
1	4	1
1	5	1
1	6	1
1	7	1
1	8	1
1	9	1
1	10	1
1	11	1
1	12	1
2	3	1
2	4	1
2	5	1
2	6	1
2	7	1
2	8	1
2	9	1
2	10	1
2	11	1
2	12	1
3	4	1
3	5	1
3	6	1
3	7	1
3	8	1
3	9	1
3	10	1
3	11	1
3	12	1
4	5	1
4	6	1

4	7	1
4	8	1
4	9	1
4	10	1
4	11	1
4	12	1
5	6	1
5	7	1
5	8	1
5	9	1
5	10	1
5	11	1
5	12	1
6	7	1
6	8	1
6	9	1
6	10	1
6	11	1
6	12	1
7	8	1
7	9	1
7	10	1
7	11	1
7	12	1
8	9	1
8	10	1
8	11	1
8	12	1
9	10	1
9	11	1
9	12	1
10	11	1
10	12	1
11	12	1

A.1.3. Demand Matrix (Unit Demand Between *Adjacent* Node Pairs)

O	D	NBUNITS
0	1	1
0	2	1
0	3	1
0	12	1
1	2	1
1	6	1
1	12	1
2	4	1
2	6	1
3	4	1
3	5	1
4	5	1
5	6	1
5	8	1
6	7	1
6	8	1
6	10	1
7	8	1
7	9	1
9	10	1
10	11	1
10	12	1
11	12	1

A.2. 20n40s1 Network

A.2.1. Topology (identical to "Net-A" in [87] (p. 11))

NODE	X	Y	SIZE
N01	183	456	3
N02	222	322	3
N03	275	163	4
N04	266	297	5
N05	403	116	4
N06	470	253	4
N07	378	241	5
N08	331	314	4
N09	476	318	5
N10	607	311	3
N11	533	410	5
N12	634	482	3
N13	513	516	4
N14	406	389	6
N15	285	437	4
N16	338	473	5
N17	433	538	3
N18	510	637	3
N19	380	549	4
N20	260	543	3

SPAN	O	D	LENGTH	UNITCOST
S01	N01	N02	139.56	139.56
S02	N01	N04	179.36	179.36
S03	N01	N20	116.181	116.181

S04	N02	N03	167.601	167.601
S05	N02	N04	50.606	50.606
S06	N03	N07	129.201	129.201
S07	N04	N03	134.302	134.302
S08	N04	N05	227.002	227.002
S09	N04	N08	67.186	67.186
S10	N05	N03	136.356	136.356
S11	N05	N06	152.506	152.506
S12	N05	N07	127.475	127.475
S13	N06	N10	148.772	148.772
S14	N06	N15	260.923	260.923
S15	N07	N06	92.779	92.779
S16	N07	N08	86.822	86.822
S17	N07	N09	124.631	124.631
S18	N08	N09	145.055	145.055
S19	N08	N15	131.32	131.32
S20	N09	N11	108.227	108.227
S21	N09	N14	99.705	99.705
S22	N10	N09	131.187	131.187
S23	N10	N12	173.118	173.118
S24	N11	N12	124.036	124.036
S25	N11	N13	107.87	107.87
S26	N12	N18	198.497	198.497
S27	N13	N17	82.97	82.97
S28	N14	N11	128.725	128.725
S29	N14	N13	166.066	166.066
S30	N14	N17	151.427	151.427
S31	N15	N14	130.173	130.173
S32	N15	N16	64.07	64.07
S33	N16	N11	204.924	204.924
S34	N16	N14	108.074	108.074
S35	N16	N20	104.805	104.805
S36	N17	N19	54.129	54.129
S37	N18	N13	121.037	121.037
S38	N19	N16	86.833	86.833
S39	N19	N18	156.984	156.984
S40	N20	N19	120.15	120.15

A.2.2. Demand Matrix (Unit Demand Between Node Pairs)

O	D	NBUNITS
N01	N02	1
N01	N03	1
N01	N04	1
N01	N05	1
N01	N06	1
N01	N07	1
N01	N08	1
N01	N09	1
N01	N10	1
N01	N11	1
N01	N12	1
N01	N13	1
N01	N14	1
N01	N15	1
N01	N16	1
N01	N17	1
N01	N18	1
N01	N19	1
N01	N20	1
N02	N03	1
N02	N04	1
N02	N05	1
N02	N06	1
N02	N07	1
N02	N08	1
N02	N09	1
N02	N10	1
N02	N11	1
N02	N12	1
N02	N13	1
N02	N14	1
N02	N15	1
N02	N16	1
N02	N17	1
N02	N18	1
N02	N19	1
N02	N20	1
N03	N04	1

```
N03   N05   1
N03   N06   1
N03   N07   1
N03   N08   1
N03   N09   1
N03   N10   1
N03   N11   1
N03   N12   1
N03   N13   1
N03   N14   1
N03   N15   1
N03   N16   1
N03   N17   1
N03   N18   1
N03   N19   1
N03   N20   1
N04   N05   1
N04   N06   1
N04   N07   1
N04   N08   1
N04   N09   1
N04   N10   1
N04   N11   1
N04   N12   1
N04   N13   1
N04   N14   1
N04   N15   1
N04   N16   1
N04   N17   1
N04   N18   1
N04   N19   1
N04   N20   1
N05   N06   1
N05   N07   1
N05   N08   1
N05   N09   1
N05   N10   1
N05   N11   1
N05   N12   1
N05   N13   1
N05   N14   1
N05   N15   1
N05   N16   1
N05   N17   1
N05   N18   1
N05   N19   1
N05   N20   1
N06   N07   1
N06   N08   1
N06   N09   1
N06   N10   1
N06   N11   1
N06   N12   1
N06   N13   1
N06   N14   1
N06   N15   1
N06   N16   1
N06   N17   1
N06   N18   1
N06   N19   1
N06   N20   1
N07   N08   1
N07   N09   1
N07   N10   1
N07   N11   1
N07   N12   1
N07   N13   1
N07   N14   1
N07   N15   1
N07   N16   1
N07   N17   1
N07   N18   1
N07   N19   1
N07   N20   1
N08   N09   1
N08   N10   1
N08   N11   1
N08   N12   1
N08   N13   1
N08   N14   1
N08   N15   1
N08   N16   1
N08   N17   1
N08   N18   1
N08   N19   1
N08   N20   1
N09   N10   1
N09   N11   1
N09   N12   1
N09   N13   1
N09   N14   1
N09   N15   1
N09   N16   1
N09   N17   1
N09   N18   1
N09   N19   1
N09   N20   1
N10   N11   1
N10   N12   1
N10   N13   1
N10   N14   1
N10   N15   1
N10   N16   1
N10   N17   1
N10   N18   1
N10   N19   1
N10   N20   1
N11   N12   1
N11   N13   1
N11   N14   1
N11   N15   1
N11   N16   1
N11   N17   1
N11   N18   1
N11   N19   1
N11   N20   1
N12   N13   1
N12   N14   1
N12   N15   1
N12   N16   1
N12   N17   1
N12   N18   1
N12   N19   1
N12   N20   1
N13   N14   1
N13   N15   1
N13   N16   1
N13   N17   1
N13   N18   1
N13   N19   1
N13   N20   1
N14   N15   1
N14   N16   1
N14   N17   1
N14   N18   1
N14   N19   1
N14   N20   1
N15   N16   1
N15   N17   1
N15   N18   1
N15   N19   1
N15   N20   1
N16   N17   1
N16   N18   1
N16   N19   1
N16   N20   1
N17   N18   1
N17   N19   1
N17   N20   1
N18   N19   1
N18   N20   1
N19   N20   1
```

A.3. Canada-A Network (variant of Canada)

A.3.1. Topology (all $c_j = 1$)

NODE	X	Y	SIZE
0	79	34	2
1	205	97	2
2	104	133	3
3	10	238	2
4	72	191	3
5	172	238	4
6	261	199	6
7	313	259	3
8	219	296	3
9	379	243	2
10	412	179	4
11	447	93	2
12	288	54	4

SPAN	O	D	LENGTH	UNITCOST
0-2	0	2	1	1
0-12	0	12	1	1
1-6	1	6	1	1
1-12	1	12	1	1
2-4	2	4	1	1
2-6	2	6	1	1
3-4	3	4	1	1
3-5	3	5	1	1
4-5	4	5	1	1
5-6	5	6	1	1
5-8	5	8	1	1
6-7	6	7	1	1
6-8	6	8	1	1
6-10	6	10	1	1
7-8	7	8	1	1
7-9	7	9	1	1
9-10	9	10	1	1
10-11	10	11	1	1
10-12	10	12	1	1
11-12	11	12	1	1

A.3.2. Demand Matrix (Unit Demand Between *Adjacent* Node Pairs)

O	D	NBUNITS
0	2	1
0	12	1
1	6	1
1	12	1
2	4	1
2	6	1
3	4	1
3	5	1
4	5	1
5	6	1
5	8	1
6	7	1
6	8	1
6	10	1
7	8	1
7	9	1
9	10	1
10	11	1
10	12	1
11	12	1

A.4. COST 239-A Network (variant of COST 239)

A.4.1. Topology (all $c_j = 1$)

NODE	X	Y	SIZE
N0	140	281	6
N1	315	350	4
N2	304	298	5
N3	356	235	5
N4	447	308	4
N5	417	161	5
N6	242	159	5
N7	250	214	5
N8	185	208	5
N9	128	143	4
N10	344	50	4

SPAN	O	D	LENGTH	UNITCOST

S1	N0	N1	1	820
S3	N0	N5	1	1090
S4	N0	N7	1	400
S5	N0	N8	1	300
S6	N0	N9	1	450
S8	N1	N4	1	820
S11	N2	N4	1	730
S12	N2	N7	1	350
S13	N3	N10	1	740
S14	N3	N4	1	320
S17	N4	N5	1	660
S18	N5	N10	1	390
S19	N5	N6	1	660
S20	N6	N10	1	760
S21	N6	N7	1	390
S22	N6	N8	1	210
S23	N6	N9	1	550
S24	N7	N8	1	220
S25	N8	N9	1	390
S26	N9	N10	1	1310

A.4.2. Demand Matrix (Unit Demand Between *Adjacent* Node Pairs)

O	D	NBUNITS
N0	N1	1
N0	N5	1
N0	N7	1
N0	N8	1
N0	N9	1
N1	N4	1
N2	N4	1
N2	N7	1
N3	N4	1
N3	N10	1
N4	N5	1
N5	N6	1
N5	N10	1
N6	N7	1
N6	N8	1
N6	N9	1
N6	N10	1
N7	N8	1
N8	N9	1
N9	N10	1

A.5. 15n30s1-A Network (variant of 15n30s1)

A.5.1. Topology (all $c_j = 1$)

NODE	X	Y	SIZE
N01	125	140	3
N02	268	55	4
N03	413	162	4
N04	525	97	3
N05	290	233	6
N06	612	218	5
N07	508	349	4
N08	572	485	4
N09	402	456	4
N10	259	325	6
N11	291	598	3
N12	292	483	4
N13	186	458	3
N14	49	421	4
N15	75	274	3

SPAN	O	D	LENGTH	UNITCOST
S01	N01	N02	1	166.355
S02	N01	N05	1	189.404
S04	N02	N03	1	180.205
S05	N02	N04	1	260.409
S06	N02	N05	1	179.354
S07	N03	N04	1	129.495
S08	N03	N05	1	142.021
S09	N03	N06	1	206.729
S10	N04	N06	1	149.03
S11	N05	N07	1	246.941
S12	N05	N10	1	97.082
S13	N05	N15	1	218.874
S14	N06	N07	1	167.263
S15	N07	N08	1	150.306
S16	N07	N09	1	150.615
S17	N08	N06	1	269.98
S18	N08	N11	1	302.87
S19	N09	N08	1	172.456
S20	N09	N10	1	193.933
S21	N10	N06	1	368.86

S22	N10	N14	1	230.903
S24	N11	N14	1	299.822
S25	N12	N09	1	113.265
S26	N12	N10	1	161.409
S28	N13	N12	1	108.908
S29	N14	N13	1	141.908
S30	N15	N14	1	149.282

A.5.2. Demand Matrix (Unit Demand Between *Adjacent* Node Pairs)

O	D	NBUNITS
N01	N02	1
N01	N05	1
N02	N03	1
N02	N04	1
N02	N05	1
N03	N04	1
N03	N05	1
N03	N06	1
N04	N06	1
N05	N07	1
N05	N10	1
N05	N15	1
N06	N07	1
N06	N08	1
N06	N10	1
N07	N08	1
N07	N09	1
N08	N09	1
N08	N11	1
N09	N10	1
N09	N12	1
N10	N12	1
N10	N14	1
N11	N14	1
N12	N13	1
N13	N14	1
N14	N15	1

A.6. 15n30s1-B Network (variant of 15n30s1)

A.6.1. Topology (all $c_j = 1$)

NODE	X	Y	SIZE
N01	125	140	3
N02	268	55	4
N03	413	162	4
N04	525	97	3
N05	290	233	6
N06	612	218	5
N07	508	349	4
N08	572	485	4
N09	402	456	4
N10	259	325	6
N11	291	598	3
N12	292	483	4
N13	186	458	3
N14	49	421	4
N15	75	274	3

SPAN	O	D	LENGTH	UNITCOST
S01	N01	N02	1	166.355
S02	N01	N05	1	189.404
S05	N02	N04	1	260.409
S06	N02	N05	1	179.354
S07	N03	N04	1	129.495
S08	N03	N05	1	142.021
S10	N04	N06	1	149.03
S11	N05	N07	1	246.941
S12	N05	N10	1	97.082
S13	N05	N15	1	218.874
S14	N06	N07	1	167.263
S17	N08	N06	1	269.98
S18	N08	N11	1	302.87
S19	N09	N08	1	172.456
S24	N11	N14	1	299.822
S25	N12	N09	1	113.265
S26	N12	N10	1	161.409
S28	N13	N12	1	108.908
S29	N14	N13	1	141.908
S30	N15	N14	1	149.282

A.6.2. Demand Matrix (Unit Demand Between *Adjacent* Node Pairs)

O	D	NBUNITS
N01	N02	1
N01	N05	1
N02	N04	1
N02	N05	1
N03	N04	1
N03	N05	1
N04	N06	1
N05	N07	1
N05	N10	1
N05	N15	1
N06	N07	1

N06	N08	1
N08	N09	1
N08	N11	1
N09	N12	1
N10	N12	1
N11	N14	1
N12	N13	1
N13	N14	1
N14	N15	1

A.7. COST 239 Network

A.7.1. Topology (from [103])

NODE	X	Y	SIZE
N0	140	281	6
N1	315	350	4
N2	304	298	5
N3	356	235	5
N4	447	308	4
N5	417	161	5
N6	242	159	5
N7	250	214	5
N8	185	208	5
N9	128	143	4
N10	344	50	4

SPAN	O	D	LENGTH	UNITCOST
S1	N0	N1	820	820
S2	N0	N2	600	600
S3	N0	N5	1090	1090
S4	N0	N7	400	400
S5	N0	N8	300	300
S6	N0	N9	450	450
S7	N1	N2	320	320
S8	N1	N4	820	820
S9	N1	N8	930	930
S10	N2	N3	565	565
S11	N2	N4	730	730
S12	N2	N7	350	350
S13	N3	N10	740	740
S14	N3	N4	320	320
S15	N3	N5	340	340
S16	N3	N7	730	730
S17	N4	N5	660	660
S18	N5	N10	390	390
S19	N5	N6	660	660
S20	N6	N10	760	760
S21	N6	N7	390	390
S22	N6	N8	210	210
S23	N6	N9	550	550
S24	N7	N8	220	220
S25	N8	N9	390	390
S26	N9	N10	1310	1310

A.7.2. Demand Matrix (from [58])

O	D	NBUNITS
N0	N1	5
N0	N2	6
N0	N3	1
N0	N4	2
N0	N5	11

N0	N6	5
N0	N7	1
N0	N8	7
N0	N9	10
N0	N10	1
N1	N2	6
N1	N3	1
N1	N4	3
N1	N5	9
N1	N6	2
N1	N7	1
N1	N8	2
N1	N9	3
N1	N10	1
N2	N3	1
N2	N4	3
N2	N5	11
N2	N6	3
N2	N7	1
N2	N8	6
N2	N9	3
N2	N10	1
N3	N4	1
N3	N5	2
N3	N6	1
N3	N7	1
N3	N8	1
N3	N9	1
N3	N10	1
N4	N5	9
N4	N6	1
N4	N7	1
N4	N8	1
N4	N9	2
N4	N10	1
N5	N6	8
N5	N7	2
N5	N8	6
N5	N9	8
N5	N10	3
N6	N7	1
N6	N8	4
N6	N9	5
N6	N10	1
N7	N8	1
N7	N9	1
N7	N10	1
N8	N9	4
N8	N10	1
N9	N10	1

A.8. Generic U.S. Network

A.8.1. Topology (from colleagues at Nortel Networks)

NODE	X	Y	SIZE
ATL	704	384	6
BOS	880	158	2
CHI	606	198	3
CLE	710	198	4
DAL	470	406	4
DEN	310	246	3
ELP	270	406	3
HOU	486	454	3
IND	630	238	4
KAN	502	254	4
LA	70	318	4
MIA	792	526	2
NYC	838	182	2
SAC	46	198	4
SEA	94	18	2
SF	18	222	2
SLC	214	206	4
TAM	742	486	3
WDC	790	230	3

SPAN	O	D	LENGTH	UNITCOST
ATL-DAL	ATL	DAL	1857	1857
ATL-HOU	ATL	HOU	1782	1782
ATL-IND	ATL	IND	1369	1369
ATL-MIA	ATL	MIA	1344	1344
ATL-TAM	ATL	TAM	983	983
ATL-WDC	ATL	WDC	1171	1171
BOS-CLE	BOS	CLE	1265	1265
BOS-NYC	BOS	NYC	450	450
CHI-CLE	CHI	CLE	861	861
CHI-IND	CHI	IND	357	357
CHI-KAN	CHI	KAN	1623	1623
CLE-IND	CLE	IND	626	626
CLE-WDC	CLE	WDC	826	826
DAL-ELP	DAL	ELP	1046	1046
DAL-HOU	DAL	HOU	448	448
DAL-KAN	DAL	KAN	1004	1004
DEN-ELP	DEN	ELP	1223	1223
DEN-KAN	DEN	KAN	1034	1034
DEN-SLC	DEN	SLC	942	942
ELP-LA	ELP	LA	1665	1665
HOU-TAM	HOU	TAM	1691	1691
IND-KAN	IND	KAN	994	994
LA-SAC	LA	SAC	646	646
LA-SF	LA	SF	855	855

LA-SLC	LA	SLC	1207	1207
MIA-TAM	MIA	TAM	466	466
NYC-WDC	NYC	WDC	390	390
SAC-SEA	SAC	SEA	1474	1474
SAC-SF	SAC	SF	182	182
SAC-SLC	SAC	SLC	1236	1236
SEA-SLC	SEA	SLC	1797	1797

A.8.2. Demand Matrix (from colleagues at Nortel Networks)

This demand matrix is proprietary to Nortel Networks and cannot be published.

A.9. 13n23s Network (variant of 15n30s1)

A.9.1. Topology

NODE	X	Y	SIZE
N02	268	55	4
N04	525	97	3
N05	290	233	6
N06	612	218	5
N07	508	349	4
N08	572	485	4
N09	402	456	4
N10	259	325	6
N11	291	598	3
N12	292	483	4
N13	186	458	3
N14	49	421	4
N15	75	274	3

SPAN	O	D	LENGTH	UNITCOST
S05	N02	N04	260.409	260.409
S06	N02	N05	179.354	179.354
S10	N04	N06	149.03	149.03
S11	N05	N07	246.941	246.941
S12	N05	N10	97.082	97.082
S13	N05	N15	218.874	218.874
S14	N06	N07	167.263	167.263
S16	N07	N09	150.615	150.615
S17	N08	N06	269.98	269.98
S18	N08	N11	302.87	302.87
S19	N09	N08	172.456	172.456
S21	N10	N06	368.86	368.86
S23	N11	N12	115.004	115.004
S24	N11	N14	299.822	299.822
S25	N12	N09	113.265	113.265
S26	N12	N10	161.409	161.409
S27	N13	N10	151.717	151.717
S28	N13	N12	108.908	108.908
S29	N14	N13	141.908	141.908
S30	N15	N14	149.282	149.282
S31	N15	N02	291.908	291.908
S32	N05	N04	271.516	271.516
S33	N02	N06	380.664	380.664

A.9.2. Demand Matrix

O	D	NBUNITS
N02	N04	9
N02	N05	4
N02	N06	1
N02	N07	7
N02	N08	3
N02	N09	4
N02	N10	6
N02	N11	2
N02	N12	2
N02	N13	2
N02	N14	5
N02	N15	2
N04	N05	1
N04	N06	2
N04	N07	1
N04	N08	6
N04	N09	3
N04	N10	1
N04	N11	9
N04	N12	8
N04	N13	4
N04	N14	5
N04	N15	4
N05	N06	3
N05	N07	8
N05	N08	8
N05	N09	7
N05	N10	4
N05	N11	6
N05	N12	8
N05	N13	1
N05	N14	8
N05	N15	9
N06	N07	5
N06	N08	9
N06	N09	8
N06	N10	9
N06	N11	6

N06	N12	7
N06	N13	4
N06	N14	2
N06	N15	3
N07	N08	1
N07	N09	5
N07	N10	6
N07	N11	1
N07	N12	2
N07	N13	1
N07	N14	8
N07	N15	4
N08	N09	3
N08	N10	6
N08	N11	6
N08	N12	6
N08	N13	9
N08	N14	5
N08	N15	9
N09	N10	1
N09	N11	8
N09	N12	8
N09	N13	9
N09	N14	4
N09	N15	7
N10	N11	4
N10	N12	2
N10	N13	5
N10	N14	7
N10	N15	7
N11	N12	9
N11	N13	2
N11	N14	5
N11	N15	7
N12	N13	5
N12	N14	5
N12	N15	6
N13	N14	1
N13	N15	7
N14	N15	3

A.10. 15n26s1 Network (variant of 15n30s1)

A.10.1.Topology

NODE	X	Y	SIZE
N01	125	140	3
N02	268	55	4
N03	413	162	4
N04	525	97	3
N05	290	233	6
N06	612	218	5
N07	508	349	4
N08	572	485	4
N09	402	456	4
N10	259	325	6
N11	291	598	3
N12	292	483	4
N13	186	458	3
N14	49	421	4
N15	75	274	3

SPAN	O	D	LENGTH	UNITCOST
S01	N01	N02	166.355	166.355
S03	N01	N15	143.024	143.024
S04	N02	N03	180.205	180.205
S05	N02	N04	260.409	260.409
S06	N02	N05	179.354	179.354
S07	N03	N04	129.495	129.495
S09	N03	N06	206.729	206.729
S10	N04	N06	149.03	149.03
S11	N05	N07	246.941	246.941
S12	N05	N10	97.082	97.082
S13	N05	N15	218.874	218.874
S14	N06	N07	167.263	167.263
S15	N07	N08	150.306	150.306
S16	N07	N09	150.615	150.615
S17	N08	N06	269.98	269.98
S18	N08	N11	302.87	302.87
S19	N09	N08	172.456	172.456
S21	N10	N06	368.86	368.86
S23	N11	N12	115.004	115.004
S24	N11	N14	299.822	299.822
S25	N12	N09	113.265	113.265
S26	N12	N10	161.409	161.409
S27	N13	N10	151.717	151.717
S28	N13	N12	108.908	108.908
S29	N14	N13	141.908	141.908
S30	N15	N14	149.282	149.282

A.10.2.Demand Matrix

O	D	NBUNITS
N01	N02	9
N01	N03	5
N01	N04	9
N01	N05	6
N01	N06	2
N01	N07	7
N01	N08	10
N01	N09	4
N01	N10	10
N01	N11	8
N01	N12	3
N01	N13	3
N01	N14	6
N01	N15	5
N02	N03	2
N02	N04	6
N02	N05	8
N02	N06	10
N02	N07	9
N02	N08	10

N02	N09	7
N02	N10	7
N02	N11	9
N02	N12	9
N02	N13	10
N02	N14	4
N02	N15	10
N03	N04	2
N03	N05	6
N03	N06	2
N03	N07	3
N03	N08	7
N03	N09	9
N03	N10	5
N03	N11	3
N03	N12	3
N03	N13	2
N03	N14	7
N03	N15	6
N04	N05	9
N04	N06	5
N04	N07	5
N04	N08	7
N04	N09	2
N04	N10	2
N04	N11	6
N04	N12	10
N04	N13	8
N04	N14	5
N04	N15	8
N05	N06	7
N05	N07	5
N05	N08	10
N05	N09	10
N05	N10	6
N05	N11	7
N05	N12	4
N05	N13	2
N05	N14	7
N05	N15	9
N06	N07	8
N06	N08	3
N06	N09	9
N06	N10	5
N06	N11	10
N06	N12	10
N06	N13	9
N06	N14	6
N06	N15	8
N07	N08	3
N07	N09	7
N07	N10	8
N07	N11	5
N07	N12	7
N07	N13	3
N07	N14	10
N07	N15	4
N08	N09	4
N08	N10	9
N08	N11	9
N08	N12	7
N08	N13	4
N08	N14	2
N08	N15	9
N09	N10	6
N09	N11	3
N09	N12	8
N09	N13	8
N09	N14	3
N09	N15	2
N10	N11	5
N10	N12	8
N10	N13	3
N10	N14	10
N10	N15	8
N11	N12	10
N11	N13	4
N11	N14	2
N11	N15	5
N12	N13	4
N12	N14	7
N12	N15	5
N13	N14	8
N13	N15	8
N14	N15	5

A.11. 12n19s Network (variant of 15n30s1)

A.11.1.Topology

NODE	X	Y	SIZE
N02	268	55	4
N04	525	97	3
N05	290	233	6
N06	612	218	5
N07	508	349	4
N08	572	485	4
N09	402	456	4
N11	291	598	3
N12	292	483	4
N13	186	458	3
N14	49	421	4
N15	75	274	3

SPAN	O	D	LENGTH	UNITCOST
S05	N02	N04	260.409	260.409
S06	N02	N05	179.354	179.354
S10	N04	N06	149.03	149.03
S11	N05	N07	246.941	246.941
S14	N06	N07	167.263	167.263
S16	N07	N09	150.615	150.615
S17	N08	N06	269.98	269.98
S18	N08	N11	302.87	302.87
S19	N09	N08	172.456	172.456
S23	N11	N12	115.004	115.004
S24	N11	N14	299.822	299.822
S25	N12	N09	113.265	113.265
S28	N13	N12	108.908	108.908
S29	N14	N13	141.908	141.908
S30	N15	N14	149.282	149.282
S31	N15	N02	291.908	291.908
S32	N05	N04	271.516	271.516
S33	N15	N05	218.874	218.874
S34	N13	N05	247.873	247.873

A.11.2.Demand Matrix

O	D	NBUNITS
N02	N04	5
N02	N05	6
N02	N06	10
N02	N07	7
N02	N08	9

N02	N09	9
N02	N11	4
N02	N12	8
N02	N13	8
N02	N14	4
N02	N15	3
N04	N05	8
N04	N06	7
N04	N07	2
N04	N08	10
N04	N09	10
N04	N11	9
N04	N12	5
N04	N13	5
N04	N14	5
N04	N15	10
N05	N06	3
N05	N07	6
N05	N08	3
N05	N09	2
N05	N11	9
N05	N12	2
N05	N13	7
N05	N14	8
N05	N15	3
N06	N07	5
N06	N08	8
N06	N09	5
N06	N11	4
N06	N12	3
N06	N13	6
N06	N14	7
N06	N15	4
N07	N08	2
N07	N09	7
N07	N11	6
N07	N12	2
N07	N13	8
N07	N14	9
N07	N15	10
N08	N09	3
N08	N11	9
N08	N12	9
N08	N13	3
N08	N14	8
N08	N15	9
N09	N11	4
N09	N12	7
N09	N13	8
N09	N14	7
N09	N15	10
N11	N12	6
N11	N13	2
N11	N14	7
N11	N15	6
N12	N13	4
N12	N14	9
N12	N15	7
N13	N14	8
N13	N15	5
N14	N15	8

A.12. NSFNET Network

A.12.1.Topology (from [83])

NODE	X	Y	SIZE
N01	9	126	3
N02	57	69	3
N03	53	193	3
N04	98	117	3
N05	156	138	3
N06	152	211	4
N07	222	118	2
N08	278	84	3
N09	327	48	4
N10	312	237	2
N11	369	9	3
N12	429	53	3
N13	425	138	3
N14	360	188	3

SPAN	O	D	LENGTH	UNITCOST
S01	N01	N02	74.518	74.518
S02	N01	N03	80.156	80.156
S03	N01	N04	89.454	89.454
S04	N02	N03	124.064	124.064
S05	N02	N08	221.508	221.508
S06	N03	N06	100.623	100.623
S07	N04	N05	61.685	61.685
S08	N04	N11	291.728	291.728
S09	N05	N06	73.11	73.11
S10	N05	N07	68.964	68.964
S11	N06	N10	162.099	162.099
S12	N06	N14	209.268	209.268
S13	N07	N08	65.513	65.513
S14	N08	N09	60.803	60.803
S15	N09	N10	189.594	189.594
S16	N09	N12	102.122	102.122
S17	N09	N13	133.056	133.056
S18	N11	N12	74.404	74.404
S19	N11	N13	140.631	140.631
S20	N12	N14	151.611	151.611
S21	N13	N14	82.006	82.006

A.12.2.Demand Matrix

O	D	NBUNITS
N01	N02	7
N01	N03	2
N01	N04	10
N01	N05	5
N01	N06	5
N01	N07	3
N01	N08	9
N01	N09	8
N01	N10	8
N01	N11	8
N01	N12	5
N01	N13	7
N01	N14	9
N02	N03	7
N02	N04	3
N02	N05	5
N02	N06	9

N02	N07	1
N02	N08	10
N02	N09	3
N02	N10	7
N02	N11	7
N02	N12	6
N02	N13	2
N02	N14	3
N03	N04	5
N03	N05	6
N03	N06	1
N03	N07	1
N03	N08	1
N03	N09	2
N03	N10	1
N03	N11	8
N03	N12	2
N03	N13	5
N03	N14	4
N04	N05	6
N04	N06	7
N04	N07	5
N04	N08	10
N04	N09	5
N04	N10	5
N04	N11	7
N04	N12	8
N04	N13	10
N04	N14	2
N05	N06	7
N05	N07	1
N05	N08	5
N05	N09	1
N05	N10	2
N05	N11	4
N05	N12	7
N05	N13	3
N05	N14	7
N06	N07	8
N06	N08	8
N06	N09	8
N06	N10	5
N06	N11	2
N06	N12	2
N06	N13	2
N06	N14	3
N07	N08	10
N07	N09	3
N07	N10	10
N07	N11	2
N07	N12	7
N07	N13	10
N07	N14	7
N08	N09	10
N08	N10	4
N08	N11	7
N08	N12	1
N08	N13	6
N08	N14	5
N09	N10	4
N09	N11	8
N09	N12	5
N09	N13	4
N09	N14	9
N10	N11	9
N10	N12	5
N10	N13	5
N10	N14	2
N11	N12	1
N11	N13	3
N11	N14	9
N12	N13	5
N12	N14	1
N13	N14	9

A.13. 19n35s1 Network (related to 20n40s1)

A.13.1.Topology

NODE	X	Y	SIZE
N01	183	456	3
N02	222	322	3
N03	275	163	4
N04	266	297	5
N06	470	253	4
N07	378	241	5
N08	331	314	4
N09	476	318	5
N10	607	311	3
N11	533	410	5
N12	634	482	3
N13	513	516	4
N14	406	389	6
N15	285	437	4
N16	338	473	5
N17	433	538	3
N18	510	637	3
N19	380	549	4
N20	260	543	3

SPAN	O	D	LENGTH	UNITCOST
S01	N01	N02	139.56	139.56
S03	N01	N20	116.181	116.181
S04	N02	N03	167.601	167.601
S05	N02	N04	50.606	50.606
S07	N04	N03	134.302	134.302
S13	N06	N10	148.772	148.772
S15	N07	N06	92.779	92.779
S16	N07	N08	86.822	86.822
S17	N07	N09	124.631	124.631
S18	N08	N09	145.055	145.055
S19	N08	N15	131.32	131.32
S20	N09	N11	108.227	108.227
S21	N09	N14	99.705	99.705
S22	N10	N09	131.187	131.187
S23	N10	N12	173.118	173.118
S24	N11	N12	124.036	124.036
S25	N11	N13	107.87	107.87
S26	N12	N18	198.497	198.497
S27	N13	N17	82.97	82.97
S32	N15	N16	64.07	64.07
S33	N16	N11	204.924	204.924
S34	N16	N14	108.074	108.074
S35	N16	N20	104.805	104.805
S36	N17	N19	54.129	54.129

S37	N18	N13	121.037	121.037
S38	N19	N16	86.833	86.833
S39	N19	N18	156.984	156.984
S40	N20	N19	120.15	120.15
S41	N03	N07	129.201	129.201
S42	N04	N20	246.073	246.073
S43	N15	N01	103.755	103.755
S44	N14	N17	151.427	151.427
N03-N08	N03	N08	161.05	161.05
N04-N08	N04	N08	67.186	67.186
N09-N06	N09	N06	65.276	65.276

A.13.2.Demand Matrix

O	D	NBUNITS
N01	N02	7
N01	N03	8
N01	N04	7
N01	N06	3
N01	N07	8
N01	N08	9
N01	N09	8
N01	N10	2
N01	N11	4
N01	N12	2
N01	N13	5
N01	N14	9
N01	N15	2
N01	N16	10
N01	N17	10
N01	N18	5
N01	N19	2
N01	N20	5
N02	N03	3
N02	N04	4
N02	N06	8
N02	N07	10
N02	N08	3
N02	N09	7
N02	N10	6
N02	N11	4
N02	N12	6
N02	N13	9
N02	N14	8
N02	N15	4
N02	N16	6
N02	N17	2
N02	N18	4
N02	N19	5
N02	N20	9
N03	N04	7
N03	N06	2
N03	N07	6
N03	N08	7
N03	N09	5
N03	N10	8
N03	N11	4
N03	N12	8
N03	N13	4
N03	N14	2
N03	N15	4
N03	N16	6
N03	N17	5
N03	N18	7
N03	N19	4
N03	N20	3
N04	N06	5
N04	N07	9
N04	N08	7
N04	N09	8
N04	N10	6
N04	N11	7
N04	N12	10
N04	N13	5
N04	N14	3
N04	N15	2
N04	N16	4
N04	N17	4
N04	N18	2
N04	N19	8
N04	N20	9
N06	N07	9
N06	N08	8
N06	N09	3
N06	N10	3
N06	N11	2
N06	N12	7
N06	N13	7
N06	N14	3
N06	N15	4
N06	N16	10
N06	N17	6
N06	N18	9
N06	N19	5
N06	N20	9
N07	N08	4
N07	N09	2
N07	N10	10
N07	N11	8
N07	N12	7
N07	N13	6
N07	N14	6
N07	N15	4
N07	N16	7
N07	N17	5
N07	N18	5
N07	N19	7
N07	N20	7
N08	N09	6
N08	N10	9
N08	N11	6
N08	N12	3
N08	N13	9
N08	N14	3
N08	N15	6
N08	N16	3
N08	N17	2
N08	N18	5
N08	N19	2
N08	N20	5
N09	N10	6
N09	N11	3
N09	N12	6
N09	N13	9
N09	N14	3
N09	N15	6
N09	N16	8
N09	N17	6
N09	N18	3
N09	N19	8
N09	N20	3
N10	N11	6
N10	N12	8
N10	N13	2
N10	N14	3
N10	N15	7
N10	N16	2
N10	N17	8
N10	N18	7
N10	N19	7
N10	N20	7
N11	N12	3
N11	N13	8
N11	N14	7
N11	N15	3
N11	N16	7
N11	N17	3
N11	N18	7
N11	N19	8
N11	N20	7
N12	N13	4
N12	N14	3
N12	N15	4
N12	N16	5
N12	N17	7
N12	N18	5
N12	N19	5
N12	N20	8
N13	N14	9
N13	N15	10
N13	N16	4
N13	N17	9
N13	N18	5
N13	N19	8
N13	N20	9
N14	N15	7
N14	N16	4
N14	N17	5
N14	N18	7
N14	N19	9
N14	N20	3
N15	N16	2
N15	N17	2
N15	N18	7
N15	N19	5
N15	N20	9
N16	N17	7
N16	N18	2
N16	N19	6
N16	N20	5
N17	N18	8
N17	N19	2
N17	N20	8
N18	N19	10
N18	N20	9
N19	N20	9

A.14. 40n80s1-60s Network

A.14.1.Topology

NODE	X	Y	SIZE
N01	300	55	3
N02	363	90	4
N03	403	56	3
N04	437	104	3
N05	399	129	3
N06	307	126	3
N07	275	168	5
N08	363	160	4
N09	470	157	5
N10	389	255	4
N11	479	216	5
N12	505	266	3
N13	358	219	5
N14	298	206	3
N15	267	226	4
N16	360	314	8
N17	524	323	3
N18	557	379	3
N19	574	448	3
N20	419	337	4
N21	253	296	4
N22	206	184	3
N23	223	334	3
N24	190	425	4
N25	309	390	6
N26	424	385	5
N27	356	429	4
N28	283	444	5
N29	218	478	6
N30	388	486	4
N31	455	446	5
N32	531	497	3
N33	434	535	3
N34	324	498	5
N35	200	549	3
N36	285	563	4
N37	380	564	4
N38	485	599	4
N39	218	591	3
N40	344	610	4

SPAN	O	D	LENGTH	UNITCOST
S01	N01	N02	72.069	72.069
S04	N02	N06	66.573	66.573
S05	N03	N04	58.822	58.822
S06	N04	N02	75.313	75.313
S07	N04	N09	62.434	62.434
S08	N05	N03	73.11	73.11
S10	N05	N09	76.322	76.322
S11	N07	N06	52.802	52.802
S12	N07	N13	97.417	97.417
S13	N08	N01	122.45	122.45
S14	N08	N07	88.363	88.363
S15	N08	N09	107.042	107.042
S17	N09	N10	127.142	127.142
S18	N10	N11	98.087	98.087
S20	N11	N12	56.356	56.356
S22	N12	N17	60.083	60.083
S24	N13	N10	47.508	47.508
S25	N13	N16	95.021	95.021
S26	N13	N21	130.208	130.208
S27	N14	N07	44.418	44.418
S28	N15	N14	36.892	36.892
S29	N15	N16	128.035	128.035
S30	N15	N21	71.386	71.386
S31	N16	N11	154.159	154.159
S33	N16	N20	63.325	63.325
S36	N17	N18	65	65
S37	N17	N38	278.742	278.742
S38	N18	N19	71.063	71.063
S40	N19	N32	65.192	65.192
S41	N20	N12	111.521	111.521

S42	N21	N25	109.417	109.417
S44	N22	N21	121.462	121.462
S45	N23	N22	150.96	150.96
S46	N24	N23	96.799	96.799
S47	N24	N25	124.04	124.04
S48	N25	N23	102.626	102.626
S49	N25	N26	115.109	115.109
S50	N25	N28	59.933	59.933
S51	N25	N29	126.59	126.59
S52	N26	N18	133.135	133.135
S53	N26	N20	48.26	48.26
S54	N26	N31	68.425	68.425
S56	N27	N31	100.449	100.449
S57	N28	N27	74.525	74.525
S58	N28	N29	73.355	73.355
S59	N28	N34	67.801	67.801
S62	N29	N36	108.231	108.231
S65	N31	N33	91.444	91.444
S66	N32	N30	143.422	143.422
S67	N32	N38	111.893	111.893
S68	N33	N37	61.294	61.294
S69	N33	N38	81.835	81.835
S70	N34	N36	75.802	75.802
S72	N35	N29	73.246	73.246
S74	N37	N30	78.409	78.409
S75	N37	N34	86.556	86.556
S76	N38	N40	141.428	141.428
S77	N39	N35	45.695	45.695
S78	N39	N36	72.615	72.615
S80	N40	N39	127.424	127.424

A.14.2.Demand Matrix

O	D	NBUNITS
N01	N02	2
N01	N03	1
N01	N04	6
N01	N05	10
N01	N06	7
N01	N07	3
N01	N08	7
N01	N09	2
N01	N10	10
N01	N11	7
N01	N12	9
N01	N13	5
N01	N14	4
N01	N15	4
N01	N16	2
N01	N17	2
N01	N18	9
N01	N19	5
N01	N20	4
N01	N21	10
N01	N22	4
N01	N23	5
N01	N24	5
N01	N25	2
N01	N26	6
N01	N27	8
N01	N28	8
N01	N29	7
N01	N30	2
N01	N31	10
N01	N32	5
N01	N33	10
N01	N34	1
N01	N35	5
N01	N36	5
N01	N37	3
N01	N38	7
N01	N39	10
N01	N40	7
N02	N03	9
N02	N04	7
N02	N05	1
N02	N06	6
N02	N07	1
N02	N08	4
N02	N09	5
N02	N10	7
N02	N11	6
N02	N12	4
N02	N13	7
N02	N14	2
N02	N15	3
N02	N16	8
N02	N17	5
N02	N18	8
N02	N19	4
N02	N20	2
N02	N21	3
N02	N22	8
N02	N23	2
N02	N24	2
N02	N25	8
N02	N26	6
N02	N27	7
N02	N28	2
N02	N29	4
N02	N30	1
N02	N31	8
N02	N32	7
N02	N33	4
N02	N34	7
N02	N35	4
N02	N36	1
N02	N37	3
N02	N38	9
N02	N39	9
N02	N40	1
N03	N04	3
N03	N05	10
N03	N06	6
N03	N07	3
N03	N08	5
N03	N09	3
N03	N10	2
N03	N11	2
N03	N12	3
N03	N13	10
N03	N14	9
N03	N15	5
N03	N16	9
N03	N17	10
N03	N18	1
N03	N19	6
N03	N20	2
N03	N21	8
N03	N22	7
N03	N23	1
N03	N24	2
N03	N25	3
N03	N26	6
N03	N27	5
N03	N28	1
N03	N29	7
N03	N30	9
N03	N31	9
N03	N32	1
N03	N33	4
N03	N34	3
N03	N35	4
N03	N36	1
N03	N37	10
N03	N38	4
N03	N39	1
N03	N40	9
N04	N05	2
N04	N06	7
N04	N07	1
N04	N08	2
N04	N09	5
N04	N10	4
N04	N11	1
N04	N12	9
N04	N13	4
N04	N14	3
N04	N15	9
N04	N16	7
N04	N17	7
N04	N18	6
N04	N19	9
N04	N20	10
N04	N21	6

N04	N22	2
N04	N23	7
N04	N24	1
N04	N25	3
N04	N26	9
N04	N27	7
N04	N28	6
N04	N29	10
N04	N30	8
N04	N31	8
N04	N32	3
N04	N33	9
N04	N34	9
N04	N35	9
N04	N36	6
N04	N37	2
N04	N38	4
N04	N39	7
N04	N40	7
N05	N06	7
N05	N07	3
N05	N08	10
N05	N09	4
N05	N10	1
N05	N11	9
N05	N12	8
N05	N13	9
N05	N14	3
N05	N15	8
N05	N16	3
N05	N17	1
N05	N18	4
N05	N19	5
N05	N20	8
N05	N21	1
N05	N22	5
N05	N23	9
N05	N24	5
N05	N25	2
N05	N26	3
N05	N27	1
N05	N28	10
N05	N29	8
N05	N30	2
N05	N31	5
N05	N32	4
N05	N33	5
N05	N34	9
N05	N35	10
N05	N36	7
N05	N37	6
N05	N38	2
N05	N39	7
N05	N40	6
N06	N07	10
N06	N08	7
N06	N09	1
N06	N10	2
N06	N11	9
N06	N12	1
N06	N13	6
N06	N14	9
N06	N15	9
N06	N16	4
N06	N17	9
N06	N18	6
N06	N19	1
N06	N20	5
N06	N21	6
N06	N22	4
N06	N23	1
N06	N24	9
N06	N25	9
N06	N26	5
N06	N27	5
N06	N28	8
N06	N29	3
N06	N30	4
N06	N31	8
N06	N32	1
N06	N33	4
N06	N34	3
N06	N35	8
N06	N36	8
N06	N37	5
N06	N38	10
N06	N39	5
N06	N40	3
N07	N08	7
N07	N09	6
N07	N10	8
N07	N11	6
N07	N12	1
N07	N13	9
N07	N14	8
N07	N15	7
N07	N16	8
N07	N17	5
N07	N18	2
N07	N19	6
N07	N20	2
N07	N21	4
N07	N22	3
N07	N23	5
N07	N24	7
N07	N25	8
N07	N26	10
N07	N27	2
N07	N28	3
N07	N29	7
N07	N30	1
N07	N31	4
N07	N32	6
N07	N33	9
N07	N34	6
N07	N35	1
N07	N36	6
N07	N37	7
N07	N38	6
N07	N39	3
N07	N40	9
N08	N09	5
N08	N10	9
N08	N11	1
N08	N12	3
N08	N13	3
N08	N14	3
N08	N15	2
N08	N16	9
N08	N17	4
N08	N18	7
N08	N19	8
N08	N20	9
N08	N21	6
N08	N22	4
N08	N23	9
N08	N24	3
N08	N25	10
N08	N26	1
N08	N27	1
N08	N28	4
N08	N29	6
N08	N30	8
N08	N31	2
N08	N32	5
N08	N33	1
N08	N34	10
N08	N35	8
N08	N36	1
N08	N37	1
N08	N38	10
N08	N39	1
N08	N40	5
N09	N10	5
N09	N11	9
N09	N12	10
N09	N13	2
N09	N14	8
N09	N15	9
N09	N16	4
N09	N17	5
N09	N18	3
N09	N19	10
N09	N20	6
N09	N21	9
N09	N22	5
N09	N23	9
N09	N24	1
N09	N25	8
N09	N26	2
N09	N27	8
N09	N28	4
N09	N29	8
N09	N30	10
N09	N31	4
N09	N32	5
N09	N33	9
N09	N34	2
N09	N35	1
N09	N36	2
N09	N37	7
N09	N38	7
N09	N39	2
N09	N40	4
N10	N11	4
N10	N12	9
N10	N13	3
N10	N14	6
N10	N15	5
N10	N16	1
N10	N17	6
N10	N18	7
N10	N19	3
[illegible]	N20	4
N10	N21	9
N10	N22	1
N10	N23	10
N10	N24	2
N10	N25	8
N10	N26	2
N10	N27	5
N10	N28	1
N10	N29	1
N10	N30	10
N10	N31	6
N10	N32	5
N10	N33	1
N10	N34	9
N10	N35	4
N10	N36	10
N10	N37	3
N10	N38	4
N10	N39	5
N10	N40	3
N11	N12	3
N11	N13	1
N11	N14	10
N11	N15	8
N11	N16	5
N11	N17	3
N11	N18	10
N11	N19	10
N11	N20	9
N11	N21	9
N11	N22	5
N11	N23	8
N11	N24	7
N11	N25	9
N11	N26	7
N11	N27	4
N11	N28	10
N11	N29	8
N11	N30	1
N11	N31	9
N11	N32	9
N11	N33	7
N11	N34	6
N11	N35	2
N11	N36	2
N11	N37	2
N11	N38	2
N11	N39	7
N11	N40	10

N12	N13	9
N12	N14	7
N12	N15	9
N12	N16	4
N12	N17	8
N12	N18	2
N12	N19	7
N12	N20	10
N12	N21	6
N12	N22	1
N12	N23	4
N12	N24	1
N12	N25	6
N12	N26	4
N12	N27	7
N12	N28	5
N12	N29	3
N12	N30	8
N12	N31	5
N12	N32	7
N12	N33	4
N12	N34	2
N12	N35	9
N12	N36	9
N12	N37	9
N12	N38	7
N12	N39	7
N12	N40	10
N13	N14	9
N13	N15	10
N13	N16	3
N13	N17	3
N13	N18	10
N13	N19	10
N13	N20	8
N13	N21	1
N13	N22	6
N13	N23	8
N13	N24	[illegible]
N13	N25	6
N13	N26	9
N13	N27	5
N13	N28	6
N13	N29	2
N13	N30	10
N13	N31	1
N13	N32	8
N13	N33	7
N13	N34	4
N13	N35	2
N13	N36	4
N13	N37	5
N13	N38	6
N13	N39	4
N13	N40	10
N14	N15	3
N14	N16	3
N14	N17	9
N14	N18	10
N14	N19	2
N14	N20	1
N14	N21	4
N14	N22	2
N14	N23	10
N14	N24	8
N14	N25	9
N14	N26	5
N14	N27	10
N14	N28	7
N14	N29	2
N14	N30	10
N14	N31	1
N14	N32	2
N14	N33	1
N14	N34	9
N14	N35	8
N14	N36	10
N14	N37	3
N14	N38	1
N14	N39	2
N14	N40	5
N15	N16	10
N15	N17	4
N15	N18	9
N15	N19	4
N15	N20	8
N15	N21	8
N15	N22	8
N15	N23	6
N15	N24	4
N15	N25	1
N15	N26	8
N15	N27	3
N15	N28	9
N15	N29	5
N15	N30	4
N15	N31	6
N15	N32	4
N15	N33	1
N15	N34	7
N15	N35	1
N15	N36	7
N15	N37	9
N15	N38	4
N15	N39	3
N15	N40	7
N16	N17	5
N16	N18	1
N16	N19	4
N16	N20	5
N16	N21	10
N16	N22	3
N16	N23	2
N16	N24	7
N16	N25	10
N16	N26	7
N16	N27	6
N16	N28	3
N16	N29	2
N16	N30	3
N16	N31	6
N16	N32	7
N16	N33	8
N16	N34	3
N16	N35	4
N16	N36	8
N16	N37	7
N16	N38	5
N16	N39	6
N16	N40	2
N17	N18	6
N17	N19	2
N17	N20	4
N17	N21	4
N17	N22	3
N17	N23	7
N17	N24	8
N17	N25	7
N17	N26	7
N17	N27	4
N17	N28	9
N17	N29	9
N17	N30	7
N17	N31	2
N17	N32	2
N17	N33	10
N17	N34	2
N17	N35	9
N17	N36	1
N17	N37	8
N17	N38	6
N17	N39	2
N17	N40	4
N18	N19	8
N18	N20	1
N18	N21	3
N18	N22	8
N18	N23	9
N18	N24	2
N18	N25	5
N18	N26	6
N18	N27	5
N18	N28	1
N18	N29	10
N18	N30	10
N18	N31	7
N18	N32	7
N18	N33	10
N18	N34	7
N18	N35	8
N18	N36	1
N18	N37	2
N18	N38	8
N18	N39	4
N18	N40	4
N19	N20	3
N19	N21	7
N19	N22	3
N19	N23	10
N19	N24	8
N19	N25	10
N19	N26	2
N19	N27	3
N19	N28	5
N19	N29	6
N19	N30	5
N19	N31	6
N19	N32	6
N19	N33	4
N19	N34	2
N19	N35	6
N19	N36	2
N19	N37	8
N19	N38	7
N19	N39	3
N19	N40	3
N20	N21	9
N20	N22	2
N20	N23	4
N20	N24	4
N20	N25	8
N20	N26	3
N20	N27	2
N20	N28	3
N20	N29	1
N20	N30	8
N20	N31	9
N20	N32	8
N20	N33	7
N20	N34	2
N20	N35	5
N20	N36	3
N20	N37	2
N20	N38	5
N20	N39	6
N20	N40	7
N21	N22	8
N21	N23	9
N21	N24	2
N21	N25	6
N21	N26	9
N21	N27	5
N21	N28	7
N21	N29	5
N21	N30	8
N21	N31	9
N21	N32	6
N21	N33	5
N21	N34	6
N21	N35	3
N21	N36	5
N21	N37	4
N21	N38	6
N21	N39	5
N21	N40	3
N22	N23	7
N22	N24	2
N22	N25	6
N22	N26	6
N22	N27	9
N22	N28	10
N22	N29	1
N22	N30	8

N22	N31	5
N22	N32	4
N22	N33	7
N22	N34	3
N22	N35	2
N22	N36	9
N22	N37	6
N22	N38	2
N22	N39	3
N22	N40	8
N23	N24	8
N23	N25	10
N23	N26	5
N23	N27	3
N23	N28	1
N23	N29	6
N23	N30	5
N23	N31	10
N23	N32	9
N23	N33	4
N23	N34	10
N23	N35	6
N23	N36	5
N23	N37	8
N23	N38	9
N23	N39	5
N23	N40	3
N24	N25	3
N24	N26	10
N24	N27	3
N24	N28	7
N24	N29	1
N24	N30	5
N24	N31	1
N24	N32	6
N24	N33	1
N24	N34	1
N24	N35	4
N24	N36	3
N24	N37	5
N24	N38	5
N24	N39	4
N24	N40	6
N25	N26	9
N25	N27	7
N25	N28	6
N25	N29	1
N25	N30	7
N25	N31	3
N25	N32	8
N25	N33	7
N25	N34	2
N25	N35	8
N25	N36	6
N25	N37	4
N25	N38	10
N25	N39	2
N25	N40	6
N26	N27	1
N26	N28	2
N26	N29	10
N26	N30	10
N26	N31	9
N26	N32	10
N26	N33	6
N26	N34	4
N26	N35	5
N26	N36	9
N26	N37	8
N26	N38	6
N26	N39	10
N26	N40	8
N27	N28	8
N27	N29	6
N27	N30	7
N27	N31	10
N27	N32	1
N27	N33	5
N27	N34	10
N27	N35	7
N27	N36	7
N27	N37	8
N27	N38	6
N27	N39	10
N27	N40	8
N28	N29	10
N28	N30	1
N28	N31	6
N28	N32	1
N28	N33	9
N28	N34	6
N28	N35	10
N28	N36	9
N28	N37	4
N28	N38	5
N28	N39	4
N28	N40	3
N29	N30	4
N29	N31	3
N29	N32	6
N29	N33	3
N29	N34	4
N29	N35	6
N29	N36	9
N29	N37	10
N29	N38	6
N29	N39	7
N29	N40	10
N30	N31	4
N30	N32	5
N30	N33	3
N30	N34	6
N30	N35	1
N30	N36	10
N30	N37	1
N30	N38	8
N30	N39	6
N30	N40	5
N31	N32	6
N31	N33	4
N31	N34	3
N31	N35	5
N31	N36	5
N31	N37	8
N31	N38	4
N31	N39	10
N31	N40	4
N32	N33	1
N32	N34	2
N32	N35	3
N32	N36	3
N32	N37	3
N32	N38	4
N32	N39	9
N32	N40	7
N33	N34	10
N33	N35	7
N33	N36	10
N33	N37	9
N33	N38	8
N33	N39	4
N33	N40	10
N34	N35	10
N34	N36	3
N34	N37	8
N34	N38	9
N34	N39	3
N34	N40	8
N35	N36	1
N35	N37	1
N35	N38	4
N35	N39	5
N35	N40	1
N36	N37	6
N36	N38	5
N36	N39	7
N36	N40	3
N37	N38	5
N37	N39	8
N37	N40	5
N38	N39	10
N38	N40	9
N39	N40	5

Appendix B. AMPL Models

B.1. *p*-Cycle SCP (implementation of model presented in [29],[40])

```
# p-cycle SCP IP Model for AMPL - Version 1.0
# 11-December-2001 by John Doucette
# Copyright (C) 2001 TRLabs, Inc. All Rights Reserved.

#****************************

# This is an AMPL model for determining the minimum-cost p-cycle network design.
# This model optimizes p-cycles only... working capacity is provided as inputs.

#****************************

# This model, including any data and algorithms contained herein, is the
# exclusive property of TRLabs, held on behalf of its sponsors. Except
# as specifically authorized in writing by TRLabs, the recipient of this
# model shall keep it confidential and shall protect it in whole or
# in part from disclosure and dissementation to all third parties.

# If any part of this model, including any data and algorithms contained
# herein, is used in any derivative works or publications, TRLabs shall be
# duly cited as a reference.

# TRLabs makes no representation or warranties about the suitability of
# this model, either express or implied, including but not limited to
# implied warranties of merchantability, fitness for a particular purpose,
# or non-infringement. TRLabs shall not be liable for any damages suffered
# as a result of using, modifying or distributing this model or its
# derivatives.

#****************************



#****************************
# SETS
#****************************

set SPANS;
# Set of all spans.

set PCYCLES;
# Set of all p-cycles.


#****************************
# PARAMETERS
#****************************
```

```
param Cost{j in SPANS};
# Cost of each unit of capacity on span j.

param Work{j in SPANS};
# Number of working links placed on span j.

param Xpi{p in PCYCLES, i in SPANS} default 0;
# Number of paths a single copy of p-cycle p provides for restoration of failure of
# span i (2 if straddling span, 1 if on-cycle span, 0 otherwise).

param pCrossesj{p in PCYCLES, j in SPANS} := sum{i in SPANS: i = j and Xpi[p,j] = 1} 1;
# Equal to 1 if p-cycle p passes over span j, 0 otherwise.
# i.e. if Xpi[p,j] = 1, then p-cycle p crosses span j.


#****************************
# VARIABLES
#****************************

var p_cycle_usage{p in PCYCLES} >=0 integer, <=10000;
# Number of copies of p-cycle p used.

var spare{j in SPANS} >=0 integer, <=10000;
# Number of spare links placed on span j.


#****************************
# OBJECTIVE FUNCTION
#****************************

minimize sparecost: sum{j in SPANS} Cost[j] * spare[j];


#****************************
# CONSTRAINTS
#****************************

subject to full_restoration{i in SPANS}:
Work[i] <= sum{p in PCYCLES} Xpi[p,i] * p_cycle_usage[p];

subject to spare_capacity_placement{j in SPANS}:
spare[j] = sum{p in PCYCLES} pCrossesj[p,j] * p_cycle_usage[p];
```

B.2. Multi-Method 1+1 APS, *p*-Cycle, and SBPP Design (Self-selected Case) (implementation of model in [21])

```
# Mixed-BiPartial IP Model for AMPL - Version 1.3
# with Bi-criteria Objective and Partial Demand Transfers
# 17-July-2003 by Anthony Sack (TRLabs) (asack@trlabs.ca)

# Model design in collaboration with Wayne D. Grover (TRLabs), Francois
# Blouin (Nortel Networks), Hadi Nasrallah (Nortel Networks), and
# John Doucette (TRLabs).
# Portions based on Shared Backup Path Protection SCP IP Model for AMPL -
# Version 1.0 and p-Cycle SCP IP Model for AMPL - Version 1.0, both by
# John Doucette (TRLabs), 6-June-2001 and 10-December-2001 respectively.

# Copyright (C) 2003 TRLabs, Inc. All Rights Reserved.

# This model, including any data and algorithms contained herein, is the
# exclusive property of TRLabs, held on behalf of its sponsors. Except
# as specifically authorized in writing by TRLabs, the recipient of this
# model shall keep it confidential and shall protect it in whole or
# in part from disclosure and dissementation to all third parties.

# If any part of this model, including any data and algorithms contained
# herein, is used in any derivative works or publications, TRLabs shall be
# duly cited as a reference.

# TRLabs makes no representation or warranties about the suitability of
# this model, either express or implied, including but not limited to
# implied warranties of merchantability, fitness for a particular purpose,
# or non-infringement. TRLabs shall not be liable for any damages suffered
# as a result of using, modifying or distributing this model or its
# derivatives.

#****************************

# This is an AMPL model for a Mixed Protection and Restoration Formulation
# with APS, p-Cycle, and SBPP options available to the solver.  Please see
# the accompanying documentation for further details on the model design.

#****************************

# METHOD, TOPOLOGY, AND DEMAND DEFINITION

param Epsilon >= 0 default 0.0001;
# A very small positive constant.

param Large >= 0 default 100000;
# A large number meant to represent infinity.

set METHODS := 1 .. 3;
# Set of restoration methods (1 = APS, 2 = p-Cycles, 3 = SBPP).

set SPANS;
# Set of spans.

param Cost{j in SPANS} >= 0 default 1;
# Cost of placing capacity on span j due to span length.

set DEMANDS;
# Set of demands.

param DemUnits{m in METHODS, r in DEMANDS} >= 0 default 0;
# Number of units for demand r that have to use method m.
```

```
#****************************

# RESTORATION ELEMENTS DEFINITION

set PCYCLES;
# Set of candidate p-cycles.

set R{r in DEMANDS};
# Set of candidate SBPP path pairs for demand r.

param T{r in DEMANDS, j in SPANS} >= 0, <= 1 default 0;
# Is span j on the shortest cycle containing the end-nodes of demand r?
# (1 if so, 0 otherwise)

param E{r in DEMANDS, j in SPANS} >= 0, <= 1 default 0;
# Is span j on the shortest route between the end-nodes of demand r?
# (1 if so, 0 otherwise)

param X{i in SPANS, p in PCYCLES} >= 0, <= 2 default 0 integer;
# Does span i have a relationship to p-cycle p?
# (2 if straddling, 1 if on-cycle, 0 otherwise)

param Theta{j in SPANS, p in PCYCLES} := sum{k in SPANS: k = j and X[j,p] = 1} 1;
# Is span j on p-cycle p?  (1 if X[j,p] = 1, 0 otherwise)

param Delta{r in DEMANDS, j in SPANS, b in R[r]} >= 0, <= 1 default 0 integer;
# Is span j on the primary side of SBPP path pair b for demand r?
# (1 if so, 0 otherwise)

param Phi{r in DEMANDS, j in SPANS, b in R[r]} >= 0, <= 1 default 0 integer;
# Is span j on the backup side of SBPP path pair b for demand r?
# (1 if so, 0 otherwise)

#****************************

# VARIABLES

var w_total{j in SPANS} >= 0 integer;
# Working capacity on span j.

var w{m in METHODS, j in SPANS} >= 0 <= Large integer;
# Working capacity on span j for method m.

var s_total{j in SPANS} >= 0 integer;
# Spare capacity on span j.

var s{m in METHODS, j in SPANS} >= 0 <= Large integer;
# Spare capacity on span j for method m.

var alpha{m in METHODS, r in DEMANDS} >= 0 <= Large integer;
# Number of units of demand r that use method m.

var n{p in PCYCLES} >= 0 <= Large integer;
```

```
# Number of unit capacity copies of p-cycle p.

var q{r in DEMANDS, b in R[r]} >= 0 <= Large integer;
# Number of units of demand r that use SBPP path pair b.

var v{r in DEMANDS, b in R[r]} >= 0 <= 1 integer;
# Do any units of demand r use SBPP path pair b?  (1 if so, 0 otherwise)

#****************************

# OBJECTIVE FUNCTION

minimize total_cost_and_slower_methods_used:
sum{j in SPANS} ((w_total[j] + s_total[j]) * Cost[j]) + Epsilon * (sum{m in METHODS, r in
DEMANDS} alpha[m,r] * m);
# Minimizes total capacity cost and as a bi-criteria addition, will prefer
# better restoration methods where the cost is equivalent.

#****************************

# CAPACITY TOTAL, RESTORABILITY, AND SBPP CONSTRAINTS

subject to working_capacity_sum{j in SPANS}:
sum{m in METHODS} w[m,j] = w_total[j];
# Working capacity is the sum of working capacity for each method.

subject to spare_capacity_sum{j in SPANS}:
sum{m in METHODS} s[m,j] = s_total[j];
# Spare capacity is the sum of spare capacity for each method.

subject to all_demand_restored{r in DEMANDS}:
sum{m in METHODS} alpha[m,r] = sum{m in METHODS} DemUnits[m,r];
# Amount of demand restored with each method must equal total demand.

subject to demand_allocation_between_methods{m in METHODS, r in DEMANDS}:
sum{o in 1..m} alpha[o,r] >= sum{o in 1..m} DemUnits[o,r];
# Demand units must be restored with their required or a better method.

subject to all_sbpp_demand_restored{r in DEMANDS}:
sum{b in R[r]} q[r,b] = alpha[3,r];
# Amount restored on all path pairs must equal total SBPP demand.

subject to only_one_path_pair{r in DEMANDS}:
sum{b in R[r]} v[r,b] = 1;
# Only one SBPP path pair can be used for each demand.

subject to demand_allowed{r in DEMANDS, b in R[r]}:
q[r,b] <= v[r,b] * Large;
# Allows demand on the SBPP path pair chosen.

#****************************

# WORKING CAPACITY CONSTRAINTS
```

```
subject to aps_working{j in SPANS}:
sum{r in DEMANDS} alpha[1,r] * T[r,j] <= w[1,j];
# APS working capacity on each span must be adequate to place full
# APS demand quantity on both sides of the shortest cycle between the
# demand portion's end-nodes (for the primary and backup paths).

subject to pcycle_working{j in SPANS}:
sum{r in DEMANDS} alpha[2,r] * E[r,j] <= w[2,j];
# p-Cycle working capacity on each span must be adequate to place full
# p-cycle demand quantity on the shortest route between the demand
# portion's end-nodes (for the working paths).

subject to sbpp_working{j in SPANS}:
sum{r in DEMANDS, b in R[r]} q[r,b] * Delta[r,j,b] <= w[3,j];
# SBPP working capacity on each span must be adequate to place full
# SBPP demand quantity on the short side of the SBPP path pair chosen
# to protect each SBPP demand portion.

#*****************************

# SPARE CAPACITY CONSTRAINTS

subject to pcycles_sufficient{i in SPANS}:
sum{p in PCYCLES} n[p] * X[i,p] >= w[2,i];
# p-Cycle set sufficient to protect all working capacity.

subject to pcycle_spare{j in SPANS}:
sum{p in PCYCLES} n[p] * Theta[j,p] <= s[2,j];
# Adequate p-cycle spare capacity to build the p-cycle set.

subject to sbpp_spare{i in SPANS, j in SPANS: i <> j}:
sum{r in DEMANDS, b in R[r]} q[r,b] * Delta[r,i,b] * Phi[r,j,b] <= s[3,j];
# Adequate SBPP spare capacity to restore all SBPP paths affected by any span failure.
```

B.3. *p*-Cycle Modular JCP Design with Path Length Constraints (implementation of model in [110])

```
################################################################################
# Model for modular JCP p-cycle design with path length restrictions.          #
# November 24, 2003 by Adil Kodian and Anthony Sack.                           #
# Copyright (C) 2003 TRLabs, Inc.  All Rights Reserved.                        #
################################################################################

################################################################################
# COPYRIGHT NOTICE                                                             #
#                                                                              #
# This model, including any data and algorithms contained herein, is the       #
# exclusive property of TRLabs, held on behalf of its sponsors.  Except        #
# as specifically authorized in writing by TRLabs, the recipient of this       #
# model shall keep it confidential and shall protect it in whole and           #
# in part from disclosure and dissementation to all third parties.             #
#                                                                              #
# If any part of this model, including any data and algorithms contained       #
# herein, is used in any derivative works or publications, TRLabs shall be     #
# duly cited as a reference.                                                   #
################################################################################

################################################################################
# DISCLAIMER                                                                   #
#                                                                              #
# TRLabs makes no representation or warranties about the suitability of        #
# this model, either express or implied, including but not limited to          #
# implied warranties of merchantability, fitness for a particular purpose,     #
# or non-infringement. TRLabs shall not be liable for any damages suffered     #
# as a result of using, modifying or distributing this model or its            #
# derivatives.                                                                 #
################################################################################

################################################################################
# ABSTRACT                                                                     #
#                                                                              #
# This model is used to design modular JCP p-cycle networks with path length   #
# restrictions.  (Note: the convention we use is that the shorter side of the  #
# p-cycle is always L, and the longer side is always R.                        #
################################################################################

################################################################################
# SETS                                                                         #
################################################################################

set MODULE_TYPES;
# Set of available module types, indexed by m.

set SPANS;
# Set of spans, indexed by i (failed) or j (surviving).

set DEMANDS;
# Set of demand relations, indexed by r.

set CYCLES;
# Set of eligible cycles, indexed by p.

set WORK_ROUTES{r in DEMANDS};
# Set of eligible working routes for each demand relation r, indexed by q.

set ALLWORKROUTES;

################################################################################
# PARAMETERS                                                                   #
################################################################################

param Large default 100000;
# A large positive constant (100000).

param ModuleCost{m in MODULE_TYPES};
# Cost of one module of type m.
```

```
param SpanCost{j in SPANS};
# Cost of span j (includes length, and other costs).

param ModuleSize{m in MODULE_TYPES};
# Capacity in STS-1 equivalents of module type m.

param DemandUnits{r in DEMANDS} default 0;
# Number of demand units in STS-1 equivalents for relation r.

param ZetaWorkRoute{j in SPANS, r in DEMANDS, q in WORK_ROUTES[r]} default 0;
# Equal to 1 if route q is through span j for relation r, 0 otherwise.

param Xpi{p in CYCLES, i in SPANS} default 0;
# Equal to 1 if p-cycle p can provide an acceptable restoration
# path for failure of span i, 0 otherwise (or can be the actual hop value) - not used.

param Xpi_L{p in CYCLES, i in SPANS} default 0;
# Equal to 1 if the L side of p-cycle p can provide an acceptable restoration
# path for failure of span i, 0 otherwise (or can be the actual hop value).

param Xpi_R{p in CYCLES, i in SPANS} default 0;
# Equal to 1 if the R side of p-cycle p can provide an acceptable restoration
# path for failure of span i, 0 otherwise (or can be the actual hop value).

param CYCLE_HOPS{p in CYCLES} default 0;
# This is the number of hops for each cycle - not used.

param pCrossesj{p in CYCLES, j in SPANS} default 0;
# Equal to 1 if cycle p lies on span j, 0 otherwise.

param SpanWiseElligibility_L{p in CYCLES, i in SPANS} default 0;
param SpanWiseElligibility_R{p in CYCLES, i in SPANS} default 0;
#dummy parameters to keep ampl happy - used to debug dat files.

param WorkHops{q in ALLWORKROUTES};

#################################################################################
# VARIABLES                                                                     #
#################################################################################

var number_modules{j in SPANS, m in MODULE_TYPES} >= 0 <= Large integer;
# Number of modules of type m placed on span j.

var work{j in SPANS} >=0 <= Large integer;
# Working capacity on span j.

var spare{j in SPANS} >= 0 <= Large integer;
# Spare capacity on span j.

var workflow{r in DEMANDS, q in WORK_ROUTES[r]} >= 0 <= Large integer;
# Number of units for relation r that use route q.

var p_cycle_usage{p in CYCLES} >= 0 <= Large integer;
# Number of unit-capacity copies of cycle p.
```

```
var p_cycle_usage_span_rest{i in SPANS, p in CYCLES} >= 0 <= Large integer;
# Number of unit-capacity copies of cycle p required for restoration of span i.

var p_cycle_usage_span_rest_l{i in SPANS, p in CYCLES} >= 0 <= Large integer;
# Number of unit-capacity copies of cycle p required for restoration of span i,
# if the L side of the cycle is used.

var p_cycle_usage_span_rest_r{i in SPANS, p in CYCLES} >= 0 <= Large integer;
# Number of unit-capacity copies of cycle p required for restoration of span i,
# if the R side of the cycle is used.

################################################################################
# OBJECTIVE                                                                    #
################################################################################

minimize total_ModuleCost: sum{m in MODULE_TYPES, j in SPANS} ModuleCost[m] * SpanCost[j]
* number_modules[j,m];
# Minimize total module cost.

################################################################################
# CONSTRAINTS                                                                  #
################################################################################

subject to demands_met{r in DEMANDS}:
sum{q in WORK_ROUTES[r]} workflow[r,q] >= DemandUnits[r];
# All demands must be routed.

subject to working_capacity_assignment{j in SPANS}:
sum{r in DEMANDS, q in WORK_ROUTES[r]} ZetaWorkRoute[j,r,q] * workflow[r,q] <= work[j];
# Enough working capacity for all demands to be routed.

subject to full_restoration{i in SPANS}:
sum{p  in  CYCLES}  (Xpi_L[p,i]  *  p_cycle_usage_span_rest_l[i,p]  +  Xpi_R[p,i]  *
p_cycle_usage_span_rest_r[i,p]) >= work[i];
# Sufficient p-cycles to restore all working capacity.

subject to spare_capacity_placement{j in SPANS}:
sum{p in CYCLES} pCrossesj[p,j] * p_cycle_usage[p] <= spare[j];
# Enough spare capacity to form all p-cycles.

subject to modular_provisioning{j in SPANS}:
sum{m in MODULE_TYPES} ModuleSize[m] * number_modules[j,m] >= work[j] + spare[j];
# Capacity to provision working and spare can be assigned only in modular units.

subject to copies_sufficient_l{i in SPANS, p in CYCLES}:
p_cycle_usage_span_rest[i,p] >= p_cycle_usage_span_rest_l[i,p];
# Must have more copies than number of paths on L side, for each failed span.

subject to copies_sufficient_r{i in SPANS, p in CYCLES}:
p_cycle_usage_span_rest[i,p] >= p_cycle_usage_span_rest_r[i,p];
# Must have more copies than number of paths on R side, for each failed span.

subject to copies_ok_for_all_failures{i in SPANS, p in CYCLES}:
```

```
p_cycle_usage[p] >= p_cycle_usage_span_rest[i,p];
# Enough copies of p-cycle p to restore every single span failure.  This will ensure
# that the number of copies of cycle p is greater than the max number required by
# any one failure.

subject to copies_zero_unacceptable_l{i in SPANS, p in CYCLES}:
p_cycle_usage_span_rest_l[i,p] <= Large * Xpi_L[p,i];
# Number of copies is zero if L side path unacceptable.

subject to copies_zero_unacceptable_r{i in SPANS, p in CYCLES}:
p_cycle_usage_span_rest_r[i,p] <= Large * Xpi_R[p,i];
# Number of copies is zero if R side path unacceptable.
```